Universitat de Lleida
Escola Politècnica Superior
Enginyeria en Informàtica

Sistemes Informàtics
Treball Final de Carrera

Determinació del subgrup de 2-Sylow d'una corba el·líptica sobre un cos binari

Autor: Albert Felip Boldú
Director: Josep Maria Miret Biosca
Maig 2008

Caminante, no hay camino, se hace camino al andar.
Antonio Machado

Índex

Índex de taules

Índex de figures

Índex de llistats

Capítol 1

Introducció

Quan llegim articles, revistes o algun *paper* que tracta el tema de les corbes el·líptiques, pot donar-se el cas que ho entenguem, ens entretingui i ens sembli realment interessant; però quins són els inicis de les corbes el·líptiques i, potser la pregunta més important, per què utilitzem les corbes el·líptiques en l'actualitat en criptografia.

La història comença al voltant del 400 AC, amb un personatge singular del qual en sabem ben poca cosa. Aquest personatge és Diofant, que treballava per aquells temps en la reconstruïda (perquè abans va ser cremada) *Biblioteca d'Alexandria.* Se sap que va existir després de l'anomenada *Edat d'or*, és a dir, després d'Euclides, Apol·loni i Arquímedes. Els matemàtics que investigaven dins la *Biblioteca* (i que havien estudiat durant 700 anys) es basaven en tres àrees: la geometria (Elements d'Euclides), la trigonometria (astronomia) i la mecànica. Això fa que el treball de Diofant sigui molt sorprenent perquè ell només estava interessat en solucionar equacions algèbriques.

Diofant va ser el primer en desenvolupar un sistema de notació algèbrica, però no va tenir cap influència en els matemàtics de l'època, però sí que en va tenir més de mil anys després amb Viète i Fermat. Diofant també va estudiar equacions algèbriques indeterminades de grau superior a 6, va treballar amb equacions quadràtiques i cúbiques, i a més a més va trobar-hi, en aquestes, solucions racionals; també va utilitzar números negatius en els seus càlculs [7].

Gairebé uns 2000 anys després de Diofant, s'ha utilitzat la seva notació i els seus principis per treballar amb corbes el·líptiques i en concret corbes el·líptiques sobre criptografia. Si busquem la paraula *criptografia* al diccionari ens diu que *és una escriptura en clau en què s'alteren els signes gràfics corresponents als fonemes d'una llengua.* Això vol dir que ho utilitzarem per a la protecció de dades a través de la xarxa, per al xifratge i per al desxifratge.

En l'actualitat, les corbes el·líptiques s'estan aplicant en la criptogra-

fia, on els primers documents interessants són el protocol de *Diffie-Hellman* [11, 6] de 1976, que és la pedra angular del que coneixem avui en dia com a criptografia de clau pública. El 1977 apareixerà un dels algoritmes més famosos fet pel Ron *Rivest* i Adi *Shamir*, anomenat *RSA* [11, 6], que és un algoritme asimètric xifrador que utilitza una clau pública. Sis anys després, el 1984, apareixia *ElGamal* [11, 6] que és un algoritme utilitzat en criptografia asimètrica que està basat en *el problema del logaritme discret*. De l'any següent, el 1985, tenim els famosos articles de *Koblitz* [9] i *Miller* [9] que van proposar els primers sistemes de criptografia basats en corbes el·líptiques. Sis anys després apareixia l'algoritme *DSA* [11, 6] de signatura i xifratge digital, això si, amb un temps de còmput molt més alt que el RSA i així podríem continuar fins avui.

A partir d'aquí la criptografia amb corbes el·líptiques ha avançat a passos de gegant, però en una recent entrevista feta a *Diffie*, *Hellman*, *Rivest* i *Shamir*, que podem anomenar com els pares de la criptografia moderna, en el context de la *RSA Conference de 2007*, tots quatre afirmen que no poden entendre que amb 30 anys la criptografia només hagi avançat el que ha avançat; ells pensaven que amb aquest període de temps hauríem obtingut sistemes segurs i fiables, i actualment no els tenim i per les persepectives de futur tardarem en tenir-los. Per acabar em quedaría amb una frase controvertida però que fa pensar extreta de l'entrevista feta a *Whitfield Diffie*: *'El pitjor que es pot dir és que la criptografia de clau pública ha sigut un gran éxit.'*[10][1]

En aquest document veurem els fonaments matemàtics necessàris per a la realització d'aquest projecte, així com els conceptes generals de corbes el·líptiques que utilitzarem per entendren el funcionament de les mateixes. Una vegada tinguem la base de la teoria de números com de corbes el·líptiques, introduïrem que és el que entenem per subgrup de 2-Sylow d'un grup i presentarem un algoritme per tal de calcular-lo en el cas del grup de punts d'una corba el·líptica sobre un cos de característica 2, seguint el disseny proposat a la tesi de *R. Moreno* [13].

Seguidament passarem a definir els objectius que volem aconseguir en aquest projecte.

[1] Si esteu interessats en llegir l'entrevista completa: Entrevista a Diffie, Hellman, Rivest i Shamir (en anglès).

1.1 Objectius

- Dissenyar segons *R. Moreno* [13] i implementar un algoritme per determinar el subgrup de 2-Sylow d'una corba el·líptica sobre un cos de característica 2.
- Distribuir totes les corbes possibles sobre un cos de característica 2 segons l'extensió del subgrup de 2-Sylow, i veure a la vegada com evolucionen quan creix l'extensió. Dins aquesta anàlisi en veurem els percentatges i en trobarem relacions. Tot acompanyat de gràfics.
- Implementar un mode de l'algoritme capaç de trobar amb corbes sobre cossos binaris grans subgrups de 2-Sylows grans (de l'ordre de 50).
- Analitzar el temps que tarda en executar-se l'algoritme per una sola corba o bé sobre un cos, per veure'n la seva eficiència.

M'he decantat per triar aquest TFC, en primer lloc per què crec que el món de la criptografia és un món ple de possibilitats i amb un nivell d'interès molt alt avui en dia ja que la seguretat és un element molt important i està molt ben valorada. Amb aquest treball tindrem la possibilitat d'entendre les corbes el·líptiques sobre cossos de característica 2, i ens permetrà fer una valoració pròpia de cap a on evolucionarà la criptografia en aquesta època d'expansió de les noves tecnologies i de la informàtica.

Agraïments

Agraeixo el suport i l'ajuda dels meus pares i del meu germà que durant tot aquest temps han estat recolzant-me en tot moment. També agraeixo als meus tiets les seves extenses correccions. Finalment donar també les gràcies al Josep Maria Miret per haver invertit en mi moltes de les seves hores i de la seva paciència. Gràcies a tots!

Capítol 2

Corbes el·líptiques

Per començar exposarem els conceptes bàsics sobre corbes el·líptiques. Farem una petita introducció a les corbes el·líptiques definides sobre un cos finit, i concretarem amb les definides sobre un cos de característica 2, en veurem algunes de les propietats i peculiaritats.

2.1 Preliminars sobre corbes el·líptiques

Una corba el·líptica sobre un cos $\mathbb{K}$ es defineix com el conjunt de punts $(x, y) \in \mathbb{K} \times \mathbb{K}$ que satisfan l'equació següent, anomenada equació general de Weierstrass:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4 + a_6$$

on els coeficients a_i són elements de $\mathbb{K}$.

Si utilitzem un canvi lineal de variables, sempre que la característica del cos sigui diferent de 2 i de 3, obtenim una equació del tipus:

$$y^2 = x^3 + ax + b,$$

on a i b $\in \mathbb{K}$ i el discriminant $\Delta = 4a^3 + 27b^2 \neq 0$. La corba sobre $\mathbb{K}$ definida per aquesta equació, anomenada equació reduïda de Weierstrass, la denotem per $E_{a,b}$ sobre un cos $\mathbb{K}$. Llavors, podem definir el conjunt de punts de la corba el·líptica $E_{a,b}$ sobre un cos $\mathbb{K}$ com:

$$E_{a,b}(\mathbb{K}) = \{(x, y) \in \mathbb{K} \times \mathbb{K} \mid y^2 = x^3 + ax + b\} \cup \left\{\mathcal{O}_{E_{a,b}}\right\}$$

on $\mathcal{O}_{E_{a,b}}$ és l'anomenat punt de l'infinit de la corba que permet dotar a aquest conjunt d'estructura de grup abelià [2].

2.1.1 Grup abelià

En primer lloc definirem que és un grup [2]. Sigui G un conjunt i $*$ una operació interna sobre G.

Es diu que $(G, *)$ és un grup, si satisfà:

1. Propietat associativa: $(g_1 * g_2) * g_3 = g_1 * (g_2 * g_3)$, $\forall\ g_1, g_2, g_3 \in G$.
2. Existeix un element neutre denotat per e tal que $g * e = e * g = g,\ \forall g \in G$.
3. Per a tot element de G existeix un element simètric, que el denotarem per g^{-1}, tal que $g * g^{-1} = g^{-1} * g = e,\ \forall g \in G$.

Un grup $(G, *)$ és diu que és abelià si satisfà:

1. Propietat commutativa: $g_1 * g_2 = g_2 * g_1,\ \forall g_1, g_2 \in G$.

2.2 Suma de punts en una corba el·líptica

Al conjunt de punts d'una corba el·líptica $E_{a,b}(\mathbb{K})$ podem definir-hi una operació suma. Aquesta operació es pot expressar gràficament pel mètode de la corba i la tangent, però també es pot expressar de forma algebraica. Aquesta operació dóna estructura de grup abelià al conjunt de punts de la corba. A continuació mostrem el mètode anteriorment nombrat.

2.2.1 Mètode de la corda i la tangent

El mètode de la corda-tangent [15] consisteix en agafar dos punts diferents P i Q de la corba, és a dir, que els punts pertanyin a $E_{a,b}(\mathbb{K})$. Tracem la recta que passi per tots dos punts. Per norma general aquesta recta tindrà tres punts de tall amb la corba el·líptica. El primer serà el punt P, l'altre serà el punt Q (ambdós ja coneguts) i el tercer serà el punt que anomenarem $-R$. Però per obtenir el punt que a nosaltres ens interessa tracem una recta vertical paral·lela a l'eix d'ordenades que passi pel punt $-R$. El nou punt intersecció d'aquesta recta amb la corba serà el punt desitjat i s'anomenarà R. Per tant es defineix la suma de P i Q com:

$$R = P + Q$$

Per entendre-ho millor mostrem, una representació gràfica:

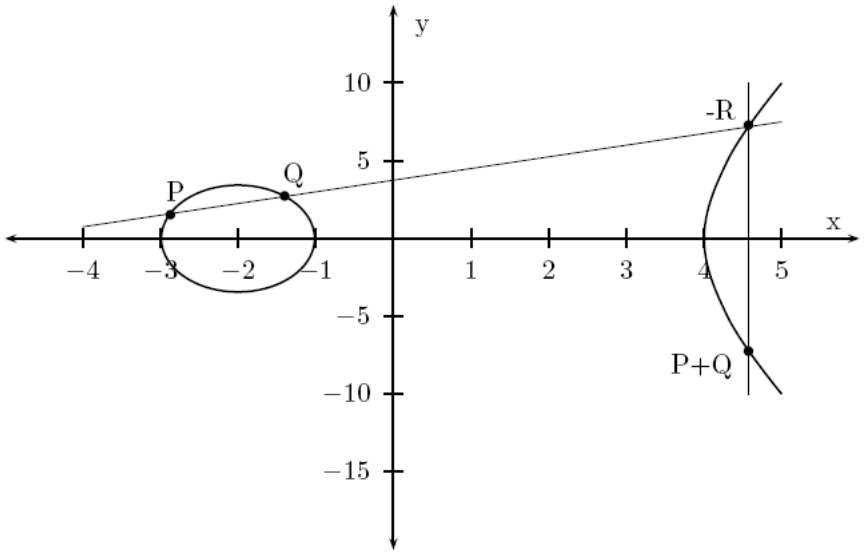


Figura 2.1: Mètode de la Corda i la Tangent quan $P \neq Q$

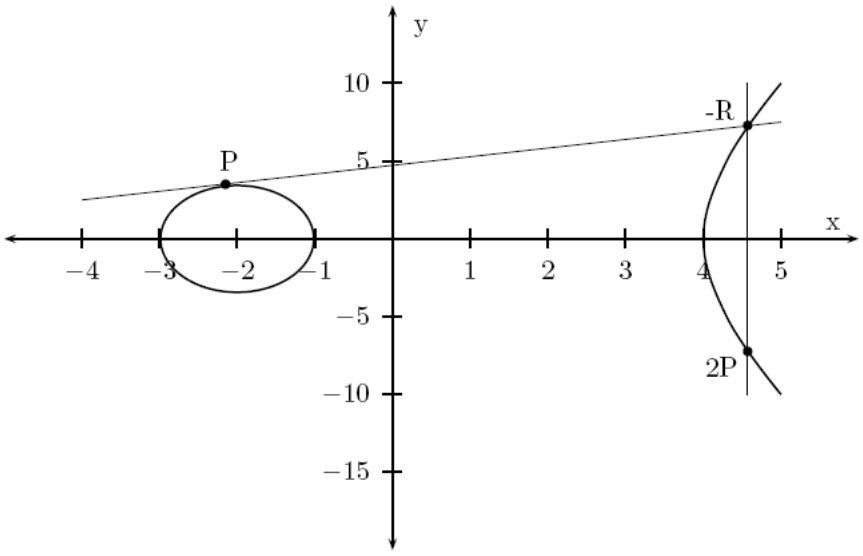


Figura 2.2: Mètode de la Corda i la Tangent quan $P = Q$

Aquest cas seria quan $P \neq Q$, però tindríem un segon cas, quan $P = Q$. En aquest cas procedirem com en l'exemple anterior, traçarem una recta tangent respecte al punt $P = Q$, que igual que abans tallarà en un tercer punt que anomenarem $-R$. A partir d'aquí traçarem una recta vertical i obtindrem el punt simètric que serà R (sempre respecte l'eix de les abscisses). Aquest nou punt R trobat serà el punt suma, denotat per $R = 2P$ (Figura 2.2), aquest serà el punt de tall amb la corba el·líptica.

El procediment que acabem de fer ara gràficament, també el podem aconseguir fer algebraicament. Siguin $P = (x_1, y_1) \in E_{a,b}(\mathbb{K})$ i $Q = (x_2, y_2) \in E_{a,b}(\mathbb{K})$, sent $E_{a,b}$ una corba el·líptica definida sobre un cos $\mathbb{K}$. Si $P \neq -Q$, llavors $P + Q = (x_3, y_3)$, on:

$$x_3 = \lambda^2 - x_1 - x_2,$$

$$y_3 = \lambda\,(x_1 - x_2) - y_1,$$

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{si} \quad P \neq Q \\ \\ \frac{3x_1^2 + a}{2y_1} & \text{si} \quad P = Q \end{cases}$$

Si $P = -Q$ aleshores $P + Q = \mathcal{O}_{E_{a,b}}$. Amb aquesta operació $(E_{a,b}(\mathbb{K}), +)$ té estructura de grup abelià amb neutre en el punt $\mathcal{O}_{E_{a,b}}$.

2.2.2 Múltiples d'un punt d'una corba el·líptica

Si $P \in E_{a,b}(\mathbb{K})$, i n és un enter, el producte nP el podem definir com segueix:

- Si $n > 0$ llavors: $\underbrace{P + P + \ldots + P}_{n \ vegades}$
- Si n=0 llavors: $\mathcal{O}_{E_{a,b}}$
- Si $n < 0$ llavors: $\underbrace{(-P) + (-P) + \ldots + (-P)}_{n \ vegades}$

En el moment de fer aquest càlcul, no és habitual fer la suma de P, n vegades, com mostra la forma anterior ja que quan la n és gran això és molt poc eficient. S'utilitza el mètode anomenat del camperol rus ja que aquest té un cost molt menor i el cost d'aquest algoritme és $O(log_2\ n)$.

Exemple 1 (Algoritme del camperol rus) *L'algoritme del camperol rus és un algoritme anomenat d'exponenciació ràpida que ens permet calcular nP. Està basat en només dues operacions la suma i la multiplicació. Aquest consisteix en sumar i multiplicar el punt P amb potències de dos. Amb això aconseguirem, com hem dit anteriorment i podem veure a continuació, que el cost del càlcul és de l'ordre $O(log_2\ n)$, que és un cost molt més baix que si fem només sumes successives (que el cost seria de $O(n)$).*

$$\begin{aligned} n &= 21_{10} = 10101_2 \\ nP &= (2^4 + 2^2 + 2^0)P \\ nP &= (2^4 + 2^2)P + P \\ nP &= (2(2^3 + 2^1)P + P) \\ nP &= (2((2^3)P + 2P) + P) \\ nP &= (2((2(2^2)P) + 2P) + P) \\ nP &= (2(2(2(2P)) + 2P) + P) \end{aligned}$$

2.3 Corbes el·líptiques sobre cossos finits

La criptografia és la pràctica i l'estudi de la informació amagada. En termes moderns la criptografia està considerada una branca de la matemàtica i de la ciència computacional i es podria dir que està estretament relacionada amb les ciències de la informació, amb la seguretat informàtica i l'enginyeria. La criptografia s'utilitza en societats tecnologicament avançades, com podrien ser els passwords o bé el comerç electrònic, tots aquests basats en criptografía.

Però en criptografia ens interessen les corbes el·líptiques, en concret les corbes el·líptiques definides sobre cossos finits, més concretament en cossos finits del tipus $\mathbb{F}_q$ amb $q = p$, p primer gran, o bé $q = 2^m$, m natural ≥ 1, aquests últims tipus de cossos són els que utilitzarem en aquest treball [3].

Una corba el·líptica $E_{a,b}$ sobre un cos $\mathbb{F}_q$, on q primer, és defineix mitjançant l'equació de la forma:

$$y^2 \equiv x^3 + ax + b \pmod{q},$$

on $a,\ b \in \mathbb{F}_q$, i on $4a^3 + 27b^2 \neq 0 \pmod{q}$. El conjunt $E_{a,b}(\mathbb{F}_q)$ està constituït per tots els punts $(x, y),\ x \in \mathbb{F}_q,\ y \in \mathbb{F}_q$, que satisfan l'equació anterior, juntament amb el punt de l'infinit $\mathcal{O}_{E_{a,b}}$.

2.3.1 Cardinal i Interval de Hasse

El cardinal d'una corba el·líptica $E_{a,b}$ sobre $\mathbb{F}_q$ que denotem per $\#E_{a,b}(\mathbb{F}_q)$, és el nombre de punts que conté la corba amb coordenades a $\mathbb{F}_q$, més el punt de l'infinit $\mathcal{O}_{E_{a,b}}$. El teorema de Hasse acota el cardinal d'una corba el·líptica.

Teorema 1 (Hasse) *Sigui $E_{a,b}$ una corba el·líptica definida sobre un cos finit $\mathbb{F}_q$, i sigui $m = \#E_{a,b}(\mathbb{F}_q)$, llavors:*

$$p + 1 - 2\sqrt{q} \leq m \leq p + 1 + 2\sqrt{q}$$

Així mateix, si $P \in E_{a,b}(\mathbb{F}_q)$, l'ordre d'aquest punt és l'enter positiu més petit n tal que $nP = \mathcal{O}_{E_{a,b}}$. Tenint en compte el teorema de Lagrange i les propietats dels grups, se satisfà:

$$\#P \,|\#E_{a,b}(\mathbb{F}_q)$$

Per calcular el cardinal d'una corba el·líptica $E_{a,b}$ sobre $\mathbb{F}_q$ podem utilitzar el mètode del recompte exhaustiu. Aquest mètode consisteix en substituir cada element de $\mathbb{F}_q$ a l'expressió x^2+ax+b i realitzar el sumatori de tots els símbols de Legendre associats als elements de $\mathbb{F}_q$ obtinguts de la substitució. Cal dir que el mètode del càlcul del cardinal pel mètode del recompte exhaustiu és ineficient i és per això que s'utilitzen altres algoritmes com per exemple el de Schoff [15] o el de Shanks-Mestre [15], que fan aquest càlcul de forma molt més efectiva, tot i que el càlcul del cardinal d'una corba és un problema computacionalment difícil [2].

2.3.2 Corbes el·líptiques sobre cossos finits de característica 2

Els cossos finits de característica 2 han de tenir aquesta forma $\mathbb{F}_q$ amb $q = 2^m$, sent m natural més gran o igual que 1. Aquests cossos són molt utilitzats en criptografía, però es caracteritzen perquè tenen unes propietats característiques molt especials que passarem a veure a continuació.

Podem utilitzar aquests cossos degut a que la paraula de dades té una longitud exacta de m bits. Això és degut a que els elements de $\mathbb{F}_{2^m}$ són representats pel conjunt de polinomis binaris de grau $\leq m-1$. En efecte, $\mathbb{F}_{2^m} = \mathbb{F}_2[x] / \langle f(x) \rangle$, on $f(x)$ és un polinomi irreductible de grau m. Aleshores cada element de $\mathbb{F}_{2^m}$ és pot veure com un polinomi:

$$a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_1 x + a_0$$

on els coeficients $a_i \in \{0, 1\}$.

Aquests polinomis es representen a l'ordenador com una paraula de m bits $a_{m-1}a_{m-2}\ldots a_1 a_0$.

Una altra forma equivalent de veure-ho és com un vector de dimensió m sobre $\mathbb{F}_2$. Llavors podrem dir que existeixen un grup de m elements $\alpha_0,\ \alpha_1,\ \ldots, \alpha_{m-1} \in \mathbb{F}_{2^m}$ i tals que cada $\alpha \in \mathbb{F}_{2^m}$ el podem escriure de la següent manera:

$$\alpha = \sum_{i=0}^{m-1} a_i \alpha_i \ on \ a_i \in \{0,1\}$$

Llavors podem representar α com un vector de 0's i 1's $(a_0, a_1, \ldots, a_{m-1})$. Si treballem amb hardware, el cos es emmagatzemat en un registre com anteriorment hem dit de m bits. A més a més la suma dels elements del cos la podem fer amb una porta lógica XOR i només ens consumirà un cicle de rellotge [8, 14].

Cal esmentar que nosaltres treballem amb LiDIA i aquesta llibreria no treballa amb la representació binària anteriorment anomenada sinó que et deixa escollir entre la representació Decimal (Dec) i la representació Hexadecimal (Hex). Nosaltres per a treballar hem elegit la representació Decimal. Només hi havia una manera de treballar amb representació binària, crear-te una funció, classe, etc... que et passés els nombres decimals a binaris. Vam creure que aquesta no era solució i no era eficient. Finament vam decidir treballar amb una de les dues representacions donades per LiDIA.

Sigui E una corba el·líptica sobre $\mathbb{F}_{2^m}$ donada per l'equació de Weierstrass:

$$E: \ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

El discriminant de l'equació de Weierstrass, que anomenarem Δ, ve donat per la següent expressió:

$$\Delta = -d_2^2 d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6, \text{ on}$$

$$\begin{aligned} d_2 &= a_1^2 + 4a_2, \\ d_4 &= 2a_4 + a_1a_3, \\ d_6 &= a_3^2 + 4a_6, \\ d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \end{aligned}$$

i el j-invariant, que anomenarem $j(E)$, ve donat per:

$$j(E) = c_4^3/\Delta,$$

$$\text{on } c_4 = d_2^2 - 24d_4$$

Si fem un canvi de coordenades podem trobar que $j(E) = (a_1)^{12}/\Delta$.
Si $j(E) \neq 0$ significa que $a_1 \neq 0$ i llavors els canvis que podem fer a les variables per obtenir una equació més simple de la corba són els següents:

$$(x, y) \rightarrow \left(a_1^2 x + \frac{a_3}{a_1}, \ a_1^3 y + \frac{a_1^2 a_4 + a_3^2}{a_1^3} \right)$$

Això ens transformarà la corba en:

$$E : y^2 + xy = x^3 + a_2x^2 + a_6,$$

on ara $\Delta = a_6$ i $j(E) = 1/a_6$.
Llavors també ens canviarien la suma de punts anteriorment definida. Ara quedaria definida així: Sigui $P = (x_1, y_1) \in E(\mathbb{F}_{2^m})$; llavors $-P = (x_1, y_1 + x_1)$. Si $Q = (x_2, y_2) \in E(\mathbb{F}_{2^m})$ i $Q \neq -P$, llavors $P + Q = (x_3, y_3)$, on:

$$x_3 = \begin{cases} \left(\frac{y_1+y_2}{x_1+x_2}\right)^2 + \frac{y_1+y_2}{x_1+x_2} + x_1 + x_2 + a_2 & \text{si} \quad P \neq Q \\ \\ x_1^2 + \frac{a_6}{x_1^2} & \text{si} \quad P = Q \end{cases}$$

$$y_3 = \begin{cases} \left(\frac{y_1+y_2}{x_1+x_2}\right) + (x_1 + x_3) + x_3 + y_1 & \text{si} \quad P \neq Q \\ \\ x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right) x_3 + x_3 & \text{si} \quad P = Q \end{cases}$$

I la segona opció que tenim és si $j(E) = 0$. En aquest cas $a_1 = 0$ i llavors els canvis que podem fer a les variables són els següents:

$$(x, y) \to (x + a_2, y)$$

Això ens transformarà la corba en:

$$E : y^2 + a_3y = x^3 + a_4x + a_6,$$

on ara, $\Delta = a_3^4$ i $j(E) = 0$.
Sigui $P = (x_1, y_1) \in E(\mathbb{F}_{2^m})$; llavors $-P = (x_1, y_1 + a_3)$. En aquest model, la suma de punts ve donada per:

Si $Q = (x_2, y_2) \in E(\mathbb{F}_{2^m})$ i $Q \neq -P$, llavors $P + Q = (x_3, y_3)$, on:

$$x_3 = \begin{cases} \left(\frac{y_1+y_2}{x_1+x_2}\right)^2 + x_1 + x_2 & \text{si} \quad P \neq Q \\ \\ \frac{x_1^4+a_4^2}{a_3^2} & \text{si} \quad P = Q \end{cases}$$

$$y_3 = \begin{cases} \left(\frac{y_1+y_2}{x_1+x_2}\right) + (x_1 + x_3) + y_1 + a_3 & \text{si} \quad P \neq Q \\ \\ \left(\frac{x_1^2+a_4}{a_3}\right) + (x_1 + x_3) + y_1 + a_3 & \text{si} \quad P = Q \end{cases}$$

En el nostre projecte només hem utilitzat un tipus d'equació que és la següent $y^2+xy = x^3+a_2x^2+a_6$ ja que busquem punts d'ordre una potència de 2. Les corbes que es poden representar amb el segon model no les estudiarem, ja que no tenen punts d'ordre 2 [11, 4].

Capítol 3

Subgrup de 2-Sylow d'una corba el·líptica sobre un cos binari

En aquest capítol veurem i desenvoluparem un procès inductiu per determinar el 2-Sylow d'una corba el·líptica sobre un cos de característica 2. El procès inductiu que aquí presentarem és una adaptació del dissenyat sobre cossos de característica p [12]. Primer veurem d'on provè el terme Sylow i que significa això.

D'on provè el nom de Sylow?

El nom de Sylow [6] provè d'un conegut matemàtic anomenat Peter Ludwig Mejdell **Sylow** (1832 - 1918). Era noruec, no rus, com molta gent es pensa. Va provar resultats fonamentals sobre grups finits. Va nèixer i morir a Christiana (ara Oslo).

Va ser professor d'escola a Frederikshald, del 1858 al 1898. Va començar de substitut a la Universitat de Christiana l'any 1862, fins que el 1898 va entrar ja com a professor; va investigar temes sobre la teoria de Galois; llavors va proposar les qüestions que el van portar a generar els teoremes de Sylow, publicats l'any 1872.

Va assistir a conferències i cursos de Chasles, Liouville, Duhammel, Kronecker i també havia d'anar a un curs de Weierstrass però en aquell moment estava malalt. De forma paral·lela també feia d'editor dels *Papers d'Abel* amb Sophie Lie. L'any 1894 va ser anomenat Honoris Causa per la Universitat de Copenhague.

3.1 Grup de 2-torsió d'una corba el·líptica

Donada una corba el·líptica E sobre $\mathbb{F}_q$ d'equació:

$$y^2 + xy = x^3 + a_2x^2 + a_6 \tag{3.1}$$

es defineix el conjunt $E[2](\mathbb{F}_q)$ com:

$$E\left[2\right](\mathbb{F}_q) = \{P \ \in \ E\left(\mathbb{F}_q\right) \mid 2P = \mathcal{O}_E\}.$$

Aquest conjunt és un subgrup de $E(\mathbb{F}_q)$, anomenat subgrup de 2-torsió de la corba, format pel punt d'ordre 2 de la corba, juntament amb el neutre (punt de l'infinit).

Per trobar el punt $P = (x, y)$ d'ordre 2, usarem el polinomi de 2-divisió de la corba, que te equació:

$$\psi_2(x) = x.$$

Per tant com que l'abscissa del punt d'ordre 2 és una arrel de $\psi_2(x)$ obtenim que $x = 0$.

Per calcular l'ordenada, substituïm el resultat de l'abscissa en l'equació 3.1 i obtenim que $y^2 = a_6$, d'on resulta que $y = \sqrt{a_6}$. Així doncs el punt inicial d'ordre 2 és:

$$P = (0, \sqrt{a_6}).$$

Més en general, per a cada $k \in \mathbb{N}$, podem definir el conjunt:

$$E\left[2^k\right](\mathbb{F}_q) = \left\{P \ \in \ E\left(\mathbb{F}_q\right) \mid 2^kP = \mathcal{O}_E\right\}$$

que té estructura de grup, i l'anomenem subgrup de 2^k-torsió de la corba. Remarquem que per a cada k es compleix que:

$$E\left[2^k\right](\mathbb{F}_q) \subseteq E\left[2^{k+1}\right](\mathbb{F}_q).$$

A més existeix un $n \in \mathbb{N}$ a partir del qual $E\left[2^n\right](\mathbb{F}_q) = E\left[2^{n+h}\right](\mathbb{F}_q)\,, \ \forall\, h \geq 1$. Per a aquesta n, el subgrup $E\left[2^n\right](\mathbb{F}_q)$ s'anomena 2-subgrup de Sylow (donat que és el màxim subgrup de $E\left(\mathbb{F}_q\right)$ de la corba que els seus punts tenen ordre una potència de 2).

En el cas de corbes sobre un cos de característica 2, com que només hi ha un punt d'ordre 2, el grup de 2-Sylow serà un grup cíclic isomorf a $\mathbb{Z}_{2^n}$.

3.2 Punts meitat de punts d'ordre 2^n

Per trobar un punt d'ordre 2^n màxim d'una corba el·líptica $E_{a,b}$ seguim un procès inductiu. Suposem l'existència d'un punt d'ordre 2^k, i determinem condicions que ens garanteixin l'existència d'un punt d'ordre 2^{k+1}. Les bases per al primer pas (punt d'ordre 2) d'aquest procès inductiu s'han fixat anteriorment.

Suposem que una determinada corba $E_{a,b}$ té punts d'ordre 2^k, on $k \geq 1$. Aquesta corba tindrà un punt $P = (x, y)$ d'ordre 2^{k+1} si, i només si, existeix un punt $Q = (\xi, \varsigma)$ d'ordre 2^k tal que $2P = Q$. Llavors diem que P és un *punt meitat* de Q. Aquesta condició és equivalent a:

$$x(2P) = \frac{x^4 + a_6}{x^2} = \xi$$

$$x^4 + \xi x^2 + a_6 = 0$$

Fent el canvi de variable $t = x^2$, obtenim l'equació:

$$t^2 + \xi t + a_6 = 0 \tag{3.2}$$

.

Si aquesta equació no té solucions, llavors no hi ha punts d'ordre 2^{k+1} i, per tant, el grup de 2-Sylow serà isomorf a $\mathbb{Z}_{2^k}$.

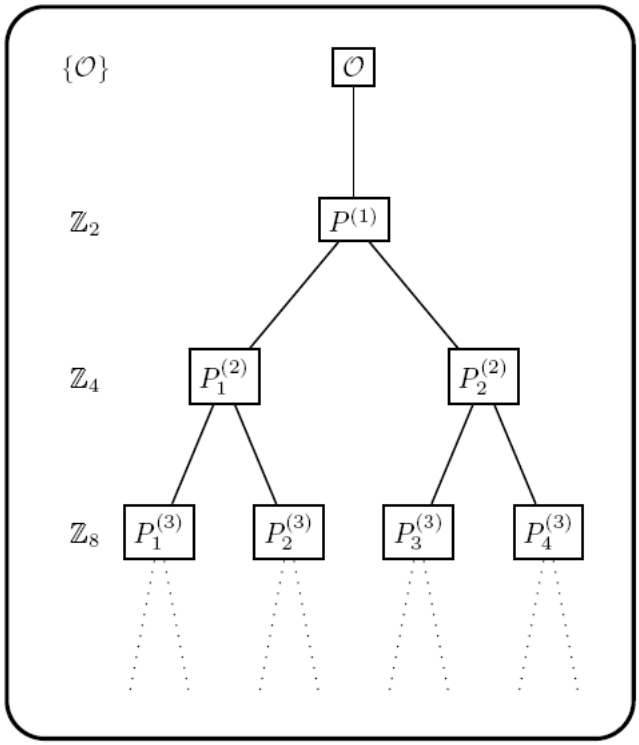
Altrament, calcularem les solucions (dues diferents o una doble) que anomenarem t_1 i t_2. Ara, per trobar el valor de la nova abscissa, només ens caldrà refer el canvi fet anteriorment, que donaria $\sqrt{t_1}$ o bé $\sqrt{t_2}$ i podem agafar qualsevol de les dues, és indiferent. En el nostre cas dins a l'aplicació utilitzem el $\sqrt{t_1}$.

En aquest punt ja tenim l'abscissa del nou punt d'ordre 2^{k+1}. Per calcular l'ordenada només ens cal substituir dins l'equació inicial de la corba 3.1 per l'abscissa ja trobada:

$$y^2 + \sqrt{t_1}y + (\sqrt{t_1})^3 + a2(\sqrt{t_1})^2 + a_6 = 0.$$

Resolent, només ens interessarà veure que l'equació té solució, llavors voldrà dir que el punt existeix. El valor obtingut en l'ordenada no ens interessarà (només ens interessarà que aquesta existeixi) [1].

Figura 3.1: Arbre de punts del subgrup de 2-Sylow.



Un factor important a tenir en compte és que ja que el subgrup de 2^k-torsió és d'ordre 2^k, l'arbre binari que es forma amb els punts d'aquest subgrup serà complet. Això fa que si en un nivell determinat un punt té un punt meitat, tots els punts del mateix nivell en tenen, havent de comprovar només l'existència d'un punt meitat per a cada nivell. Aquesta propietat és la que dóna el caràcter lineal a l'algoritme. Les figures 3.1 i 3.2 ens ho expliquen d'una forma gràfica.

Figura 3.2: Arbre d'abscisses del subgrup de 2-Sylow.

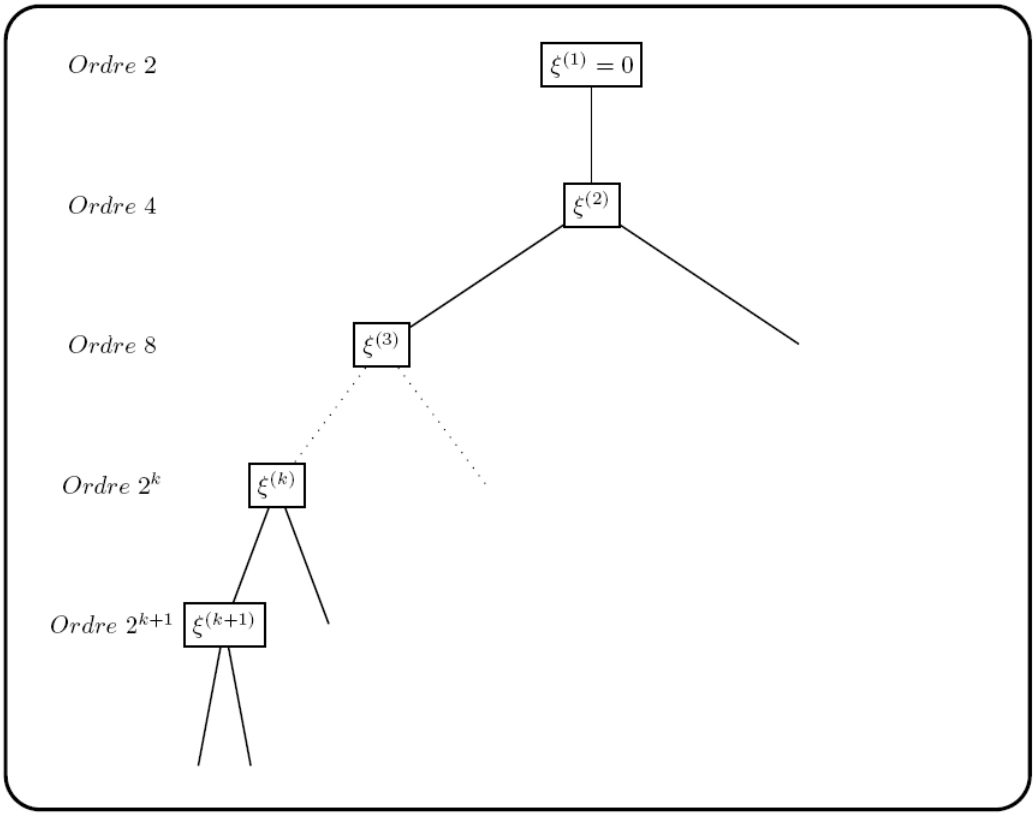
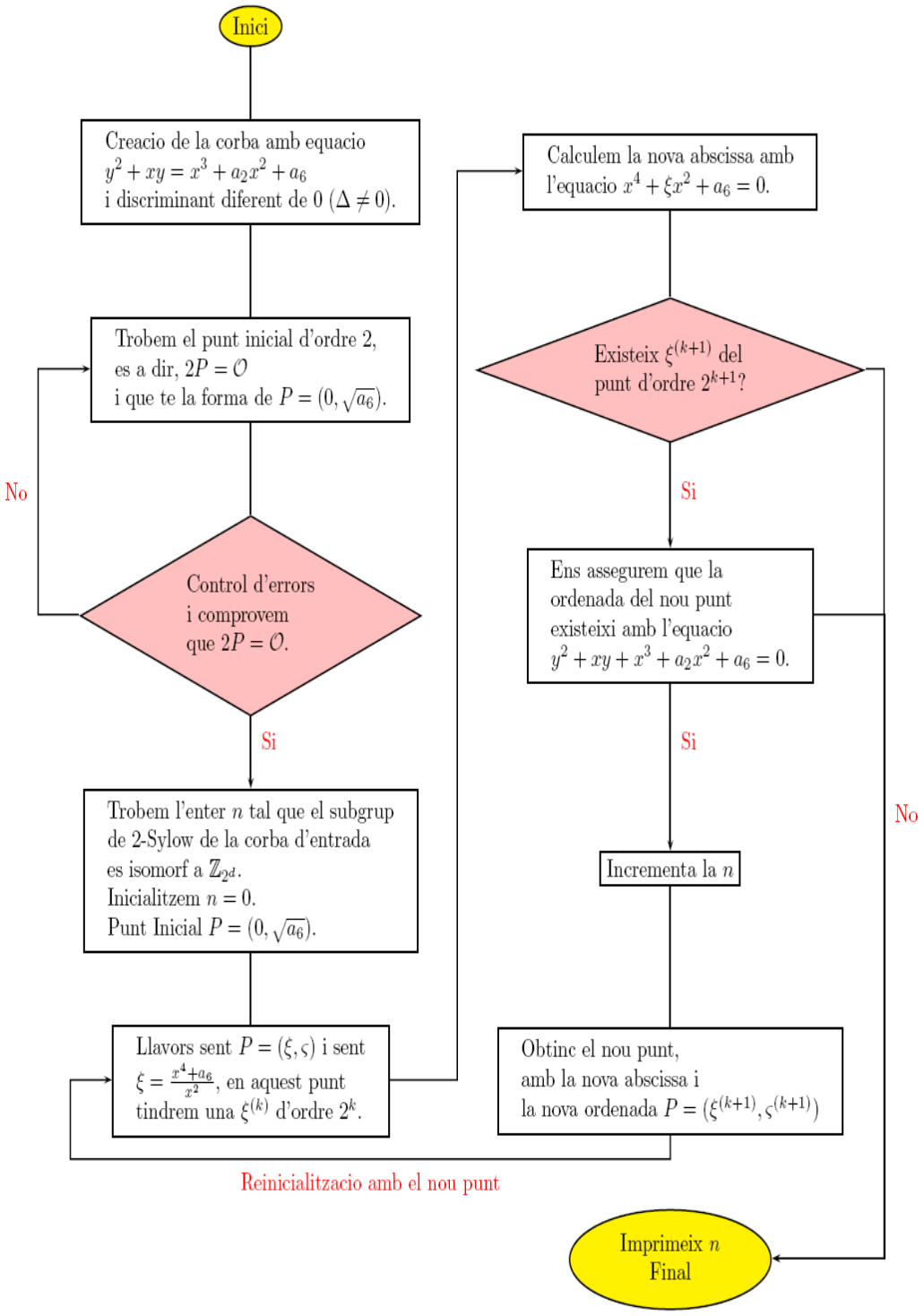


Figura 3.3: Diagrama de flux de l'algoritme implementat



Capítol 4

Implementació

La implementació del nostre algoritme ha estat realitzada en el llenguatge C++. Per al disseny i compilació s'ha utilitzat Eclipse 3.2 i g++, d'aquesta forma ens assegurem un alt grau de portabilitat cap a altres sistemes. També hem utilitzat la llibreria LiDIA [5], la versió (versió 2.2.0), perquè tenia definides les funcions i estructures necessàries per a la realització i la implementació d'aquest algoritme

Ara anem a fer un breu comentari sobre les estructures que ens ha proporcionat LiDIA i que s'han utilitzat per a aquest treball:

- **bigint:** Ens proporciona una aritmètica entera sense límit (o gairebé sense límit) en el tamany de l'element que s'utilitzarà.
- **galois_field:** És una funció que ens crearà un cos finit del tipus $\mathbb{F}_q$.
- **gf_element:** Ens permet representar els elements del cos de Galois, i poder fer càlculs amb aquests dins d'aquest cos.
- **elliptic_curve:** La utilitzarem per a la representació de les corbes el·líptiques, en el nostre cas definides sobre un cos.
- **point:** Conté la interfície per instanciar i operar sobre punts d'una corba declarada amb la classe elliptic_curve. Ofereix les operacions bàsiques que necessitem per treballar amb punts.
- **base_vector:** Ens permet emmagatzemar elements del tipus gf_element per després poder-hi treballar.
- **polynomial:** Ens proporciona una eina per crear polinomis, ja que moltes funcions de elliptic_curve necessiten polinomis com a entrada.

- **timer:** Estructura que ens permet calcular el temps que tarda el nostre algoritme en trobar alguna de les solucions vàlides.

4.1 LiDIA: La llibreria matemàtica

En aquest treball hem utilitzat la llibreria matemàtica LiDIA [5] per a la implementació del codi, ja que té definides d'una forma més o menys eficient totes les funcions sobre corbes el·líptiques que s'han necessitat per a la realització d'aquest projecte.

LiDIA és una llibreria matemàtica pensada per a càlculs de teoria de nombres desenvolupada inicialment l'any 1994 pel departament de Ciències de la Computació de la Universitat de Saarlandes, Alemanya. Actualment ha esdevingut un projecte de software lliure, però la coordinació continua a càrrec del departament. LiDIA ve recopilada en diferents paquets, aportant cadascú diferents funcionalitats i temàtiques.

En aquest projecte s'ha utilitzat un nombre concret de funcions de la llibreria però observant el manual es pot veure que és una de les llibreries més completes en referència a qualsevol aspecte de la matemàtica. En tenir implementades gran quantitat de funcions matemàtiques la converteixen en una llibreria òptima per treballar en el món de la teoria dels nombres passant per camps més aplicats, com el de la criptografia. A més té implementat un gran nombre de funcions i no només estructures bàsiques.

També en realitzar el projecte he trobat alguna que altra errata, certament important, que m'induïa a generar errors. Quan em disposava a calcular el cardinal de la corba per després fer la *valoració 2-àdica*, algunes vegades em donava un cardinal diferent amb els mateixos coeficients de la corba, després d'executar-ho dues o tres vegades seguides. Aquest error provè del fet que LiDIA al moment de calcular el cardinal es basa en una llavor aleatòria i al fer-ho moltes vegades seguides no ho feia correctament. Per resoldre aquest error vam decidir escriure a la llista de desenvolupadors de LiDIA per veure si en una versió posterior poden trobar-hi solució (John Cremona ens va contestar i ens va dir que sí, que podia ser un bug).

Un altre aspecte que pot causar confusió és que els operadors de suma, resta, multiplicació, divisió, etc... són només de dos operands, això vol dir que si l'operació és simple ens anirà molt bé, però si l'operació és més complexa ho haurem de subdividir en operacions més petites cosa que pot ser que ens causi un cert grau de confusió. A part cal dir que si la fórmula és complexa haurem d'utilitzar variables auxiliars amb els inconvenients que això comporta.

4.2 Algoritme

L'algoritme implementat té 5 modes d'execució que passarem a comentar seguidament:

- **Mode execució 1:** El primer dels modes d'execució serà una execució sense introduir-li cap argument, només fent l'execució del Makefile i l'execució del programa. Aquesta opció ens escollirà a l'atzar un número entre 1500 i 2000; el número elegit a l'atzar serà el grau o l'extensió del nostre cos (la base sempre serà 2). I a partir d'aquí ja crearem la corba i en calcularem el grup de 2-Sylow d'aquesta.
  - # *./nom_programa*
- **Mode execució 2:** Al segon dels modes d'execució li haurem d'introduir un paràmetre que passarà a ser l'extensió del nostre cos, a diferència de l'anterior que aquest número l'elegiem a l'atzar. A partir d'aquí ja calcularem també la nostra corba amb els coeficients d'aquesta elegits a l'atzar.
  - # *./nom_programa extensio*
- **Mode execució 3:** Al tercer dels modes d'execució li hem d'introduir la paraula *calcular_corbes*, una extensió i ens calcularà totes les corbes que existeixen per a aquell cos. Finalment ens donarà un recompte amb el número de corbes que tindrem per a cada n, sent l'enter tal que el subgrup de 2-Sylow és isomorf a $\mathbb{Z}_{2^n}$. Aquest mode d'execució està dissenyat per a nombres no gaire grans (fins a un grau de 19), ja que quan passem de 19 ja hem de comptabilitzar molts bilions de corbes i tarda molt de temps.
  - # *./nom_programa calcular_corbes extensio*
- **Mode execució 4:** Al quart dels modes d'execució li haurem d'introduir la paraula *calcular_concreta*, li introduirem també una extensió i un enter subgrup de 2-Sylow n (2-Sylow). El programa s'executarà fins a trobar una corba amb un n com el que nosaltres li hem introduït al cridar-lo, és a dir, isomorf a $\mathbb{Z}_{2^n}$. Si troba la corba, la imprimirà i acabarà l'execució, si no la troba continuarà executant-se fins a analitzar totes les corbes existents en aquella extensió. Si acabés i no la trobés ens imprimiria un resum per veure que per la n que hem introduït no hi ha cap corba. És un mode d'execució que ens servirà per a corbes petites, fins a un grau de 19.

  - # *./nom_programa calcular_concreta extensio 2-sylow*

- **Mode execució 5:** El cinquè dels modes d'execució és potser el més interessant. És un mode dissenyat especialment per al càlcul de Sylows grans, és a dir a partir de 50, cosa que el mode 4 abans no ens suportava (el mode 4 estava especialment dissenyat per calcular totes les corbes d'una extensió donada). Aquest mode està basat en l'aleatorietat de l'elecció dels paràmetres de la corba. Li hem d'introduir en primer lloc l'extensió del cos, després el Sylow gran i per últim el número de corbes que volem analitzar ja que al treballar amb nombres tan i tan grans no podem pretendre que ens analitzi totes les corbes sino només unes quantes.

  - # *./nom_programa calcular_concreta_v2 extensio 2-sylow n_corbes*

- **Mode error:** Si intentem fer alguna cosa no prevista en el codi ens saltarà el mode d'error, que serà una ajuda per veure com s'executen cadascún d'aquests modes.

4.2.1 Funcions principals de l'algoritme

En primer lloc explicarem els aspectes més tècnics del treball. Intentarem incidir en els detalls més concrets del disseny d'aquest, com les funcions creades i els passos generals que realitza el programa amb pseudocodi.

Després passarem a comentar les funcions creades més importants del nostre algoritme implementat, juntament amb el pseudocodi:

- **Creació de la corba (definir_corba):** En aquesta funció li hem de passar un paràmetre que serà l'extensió del nostre cos, a partir d'aquí crearem el cos de Galois, seguidament tenim l'opció d'imprimir el número d'elements del cos i el polinomi irreductible d'aquest cos. Aquesta funció serà la que ens comprovarà si el discriminant és 0 i seguidament ens crearà la corba corresponent amb dos coeficients aleatoris. A continuació continuarem calculant els punts que nosaltres desitgem de la corba.

  Paràmetres d'entrada: grau d de l'extensio del cos $\mathbb{F}_{2^d}$.
  Paràmetres de retorn: una corba el·líptica amb equació $y^2 + xy = x^3 + a_2x^2 + a_6$ amb discriminant no nul sobre el cos $\mathbb{F}_{2^d}$.

Listing 4.1: Pseudocodi de la funció Creació de la corba

```
crea_cos[1](2, extensio);
imprimeix(numero_elements_del_cos);
imprimeix(polinomi_irreductible);

inici;

a2.aleatori();
a6.aleatori();

si a6:=0 llavors ves a inici;
        sino
                e:=crearcorba(a2, a6);
                controlem_errors(a2, a6);
        fsino
fsi
```

- **Trobar el punt inicial (calcular_punt_inicial):** Li hem de passar el cos de Galois creat, la corba el·líptica ja creada (amb els dos coeficients utilitzats en la generació anterior de la corba). Aquesta estructura crea el punt d'ordre 2 de la corba creada i comprova que tingui ordre 2. Abans i després de crear el punt realitzarem una sèrie de comprovacions per constatar que el punt està ben creat i que realment és el punt que desitgem, no un altre.

  Paràmetres d'entrada: grau de l'extensió del cos de Galois ($\mathbb{F}_{2^d}$), i una corba el·líptica d'equació $y^2 + xy = x^3 + a_2x^2 + a_6$ sobre el cos $\mathbb{F}_{2^d}$.
  Paràmetres de retorn: Ens calcularà el punt d'inici d'ordre 2 de la corba el·líptica entrada.

Listing 4.2: Pseudocodi de la funció Trobar el punt inicial

```
x:=0;
y:=sqrt(a6);

crear_punt(x,y,corba);

si 'punt pertany corba' llavors
        seguim i imprimeix(punt-corba);
```

[1]Les paraules clau estan explicades en la secció 4.2.5, només ens cal clicar-hi a sobre per accedir-hi.

```
        sino
        no seguim i imprimeix(punt-NO-corba);
        fsino
fsi

comprovem(2*P):=Punt de l'Infinit;
ordre(P):=2;
```

- **Calcula la nova abscissa (calcula_abscissa):** En aquesta funció ens calcularà la nova abscissa d'un punt d'ordre 2^{k+1} a partir d'un punt donat d'ordre 2^k. Per fer això utilitzarem el polinomi que dóna aquesta nova abscissa. Crearem un vector per a les arrels del nostre polinomi i després trobarem les solucions de la nostra equació que serà la nova abscissa que utilitzarem. Retornarem qualsevol dels dos roots obtinguts. Si fos un cas que no tinguessim solució la funció no retornaria res, al detectar que no retorna res no es compliria una de les condicions que es desitgen i per tant passariem a analitzar una altra corba.

  Paràmetres d'entrada: El cos de Galois $\mathbb{F}_{2^d}$, la corba el·líptica ja creada amb equació $y^2 + xy = x^3 + a_2x^2 + a_6$ i l'abscissa d'un punt P d'ordre 2^k per calcular la nova.
  Paràmetres de retorn: L'abscissa, si existeix, d'un punt Q d'ordre 2^{k+1} tal que $2Q = P$.

Listing 4.3: Pseudocodi de la funció Calcular abscissa

```
base_vector v[3];
quadrat:=1;

inserta(a6,0);
inserta(abscissa,1);
inserta(quadrat,2);

//Creo el polinomi a partir del vector amb l'equació 3.2
poli_abs(v);

//Calculem els roots del polinomi
vector_roots:=find_roots(poli_abs);

retorna vector_roots[0];
```

- **Càlcul de la nova ordenada (calcula_equacio_ordenada):** Ens calcularà cada cop i cada volta que faci el bucle, com diu el nom, la nova ordenada del nou punt i quan l'hagi calculada ens retornarà el valor.

  Paràmetres d'entrada: el cos de Galois $\mathbb{F}_{2^d}$, la corba el·líptica ja creada amb la seva corresponent equació, els coeficients a2 i a6 de la nostra corba i la nova abscissa.
  Paràmetres de retorn: retorna un element enter que serà la nova ordenada.

  Listing 4.4: Pseudocodi de la funció Calcular equació ordenada

```
var:=y^2+t1*y+t1^3+a2*t1^2+a6;
retorna var;
```

- **Verificació de l'existència de l'ordenada (calcula_ordenada):** Ens mirarà cada volta al bucle si, i només si, existeix la nova ordenada i ens retornarà veritat o fals depenent de si existeix o no. Això ho fem perquè hi ha vegades que no ens interessarà el valor de l'ordenada, només ens interessarà si existeix o no. Dins d'aquesta funció ens cridarà a la funció anterior que ens calculava l'ordenada.

  Paràmetres d'entrada: el cos de Galois $\mathbb{F}_{2^d}$, la corba el·líptica ja creada amb la seva corresponent equació, els coeficients a2 i a6 de la nostra corba, i la nova abscissa.
  Paràmetres de retorn: retorna un booleà dient si existeix o no l'ordenada per aquesta abscissa.

  Listing 4.5: Pseudocodi de la funció Calcular ordenada

```
retorn_calcul:=calcula_equacio_ordenada(F,e,a2,a6,
    t1);
retorna boolea(retorn_calcul);
```

- **Trobar el subgrup de 2-Sylow d'una corba el·líptica:** La funció **trobar_n_maxima** ens trobarà l'enter n que determina el subgrup de 2-Sylow de la nostra corba seguint els procés inductiu explicat en les pàgines anteriors. Serà la funció més important dins l'aplicació i serà sobre la que treballaran totes les altres. Aquesta quan hagi acabat de calcular ens retornarà l'enter n que serà el 2-Sylow. Dins d'aquesta funció hem hagut de disenyar-ne altres funcions auxiliars per realitzar

els càlculs; com podrien ser *calcula_equació_ordenada*, *calcula_ordenada* i *calcula_abscissa* o bé trobar el punt d'inici[2]. Hem decidit fer-ho amb diferents funcions ja que és una manera fàcil de trobar errors, a més a més així per cada pas que realitzem tindrem una funció.

Paràmetres d'entrada: el cos de Galois $\mathbb{F}_{2^d}$, la corba el·líptica ja creada amb equació $y^2 + xy = x^3 + a_2x^2 + a_6$ i el punt inicial trobat i creat (només el punt inicial).
Paràmetres de retorn: ens trobarà l'enter n tal que el subgrup de 2-Sylow és isomorf a $\mathbb{Z}_{2^n}$, de la corba entrada.

Listing 4.6: Pseudocodi de la funció Trobar n màxima

```
enter n:=0;
punt_d'ordre_2(abscissa,ordenada);

mentre calcula_abscissa(F,e,abscissa,a6) i
    calcula_ordenada(F,e,ordenada,a2,a6,abscissa) i
    id:=1 llavors

t1:=calcula_abscissa(F,e,abscissa,a6);
t1:=sqrt(t1);
r_ordenada:=calcula_ordenada(F,e,ordenada,a2,a6,t1)

          si calcula_abscissa(F,e,abscissa,a6) i
             r_ordenada:=1 llavors
                    abscissa:=t1;
                    r_ordenada:=calcula_ordenada(F,e,
                       ordenada,a2,a6,abscissa);
                    incrementa n;
                    imprimeix(n);
                    id:=1
          sino
                    imprimeix(n);
                    id:=0;
          fsino
          fsi

fmentre
```

[2]Les funcions calcula_equació_ordenada, calcula_ordenada, calcula_abscissa i trobar el punt d'inici són les funcions que s'han explicat anteriorment.

4.2.2 Altres funcions (I)

Aquestes funcions que hem fet fins ara són les funcions bàsiques per calcular el subgrup de 2-Sylow, però com que hem ficat molts altres modes d'execució necessitarem altres funcions que es basaran en aquestes anteriors, per fer-ne de noves.

- **Calcula el subgrup de 2-Sylow de totes les corbes sobre un cos donat (calcular_totes_les_corbes):** Aquesta funció és una mica peculiar ja que tindrem dues maneres d'executar-la.

  La primera de les maneres és introduint-li un sol paràmetre que serà el grau de l'extensió del cos, aquesta funció farà els còmputs necessàris i ens imprimirà una taula, amb els diferents 2-Sylows existents depenent del paràmetre d'entrada (és a dir, de l'extensió del cos). Correspon al mode 3 explicat en la secció 4.2. En resum aquesta funció ens farà un recompte de tots els 2-Sylows de totes les corbes que puguin existir i que tinguin un punt d'ordre 2, en una extensió donada. A més a més també en calcularem el temps que tarda a fer-ho. S'ha de dir que aquesta funció utilitza vectors i ha de recórrer tots els elements del cos, per això a la màxima extensió que hem arribat és de 19 i ens ha tardat un número bastant elevat d'hores d'execució.

  La segona de les maneres és introduint-li dos paràmetres, el primer serà el grau de l'extensió del cos i el segon el 2-Sylow a buscar (no pot ser gaire gran); la funció farà els còmputs necessàris i ens imprimirà la corba amb el 2-Sylow corresponent, si fos el cas que per a aquella corba el 2-Sylow no existís ens imprimiria, en acabar, una taula resum de tot el que ha trobat i llavors nosaltres podrem comprovar que no té 2-Sylow per aquella extensió. Correspon al mode 4 d'execució explicat en la secció 4.2.

  Funció executada amb un sol paràmetre:
  Paràmetres d'entrada: el grau de l'extensió del cos.
  Paràmetres de retorn: la distribució del nombre de corbes segons el subgrup de 2-Sylow.

  Funció executada amb dos paràmetres:
  Paràmetres d'entrada: el grau de l'extensió del cos i un enter n (no gaire gran) que serà el subgrup de 2-Sylow a $\mathbb{Z}_{2^n}$ del que volem trobar una corba si existeix.
  Paràmetres de retorn: una corba sobre $\mathbb{F}_{2^d}$ tal que el subgrup de 2-Sylow és isomorf a $\mathbb{Z}_{2^n}$, si no trobés la corba ens imprimiria la distri-

bució del nombre de corbes segons el subgrup de 2-Sylow.

Listing 4.7: Pseudocodi de la funció Calcular totes les corbes

```
enter n:=0,j,z;
tamany:= 2^d;
final:=0;
vector_elements[tamany];

crea_cos(2,degree);
b:=busca_generador(F);

per a j:=0 fins a tamany-1 fer incrementa(j)
   llavors
        b:=b*a;
        inserta(vector_elements[b],j);
fper a

inserta(vector_elements[final],tamany-1);

per a z:=0 fins a numero_elements_cos fer
        per a j:=0 fins a numero_elements_cos fer
                a2:=0;
                a6:=0;

                a2:=vector_elements[z];
                a6:=vector_elements[j];
                si a6:=0 llavors
                        n:=0;
                        incrementa(z);
                        incrementa(j);
                sino
                crea_corba(a2,a6);
                n:=calcular_punt_inicial(F,e,a2,a6)
                    ;
                        si n:=p1 llavors
                        imprimeix(corba trobada);
                        surt;
                        fsi
                        incrementa(z);
                        incrementa(j);
```

```
                    fsino
                    funcio_resum(n);
                    n:=0;
                    fsi
            fper a
fper a

funcio_imprimeix(degree);
```

- **Distribució de les corbes (funcio_resum):** És la funció que va íntimament relacionada amb la funció de calcular totes les corbes i és la que ens fa un recompte de cada n obtinguda.

  Paràmetres d'entrada: enter n que serà el subgrup de 2-Sylow.
  Paràmetres de retorn: es l'encarregada de fer el resum dels 2-sylow i se'l queda en memòria.

- **Impressió de les corbes (funcio_imprimeix):** Ens imprimeix els resultats.

  Paràmetres d'entrada: el grau de l'extensió del cos.
  Paràmetres de retorn: es l'encarregada de treure els resultats per pantalla.

4.2.3 Altres funcions (II)

- **Búsqueda de 2-Sylows grans (aleatori):** Aquesta funció s'executarà quan utilitzem el mode d'execució 5 i ens servirà per trobar 2-Sylows grans, de l'ordre de 50 o més grans. En aquest tipus d'execució tenim una limitació per què que treballem amb nombres molt molt grans i no podrem analitzar totes les corbes sinó que només analitzarem el número de corbes que li entrem per paràmetres. Ens basarem en analitzar paràmetres aleatoris fins a trobar-ne uns de vàlids que satisfacin les condicions imposades. Pot passar que nosaltres realitzem l'anàlisi i no trobi cap corba que compleixi els requisits, llavors haurem de reexecutar l'aplicació.

  Paràmetres d'entrada: El grau de l'extensió del cos, l'enter n que determina el subgrup de 2-Sylow que ens interessa buscar i el número de corbes a analitzar.
  Paràmetres de retorn: Quan trobi la corba amb el 2-Sylow desitjat, l'imprimirà, si no la troba imprimirà un missatge quan hagi acabat

d'executar el número de corbes entrat per paràmetres i ens dirà que no l'ha trobat i que reexecutem l'aplicació.

Listing 4.8: Pseudocodi de la funció Aleatori

```
enter n:=0,x;

crea_cos F(2,extensio);

per a x=0 fins a x=numero_corbes fer
        inici:

        aleatori(a2);
        aleatori(a6);

        si(a6:=0)llavors ves a inici;
        sino
                crea_corba(a2,a6);
                n:=calcula_punt_inicial(F,e,a2,a6);
                si n:=sylow llavors exit(0);
                fsi
        fsino
        fsi
fper a
```

4.2.4 Funció complementària

Hem implementat aquesta funció (que la podem activar o desactivar segons les nostres preferències) i ens servirà per quan treballem amb números molt grans per comprovar si el subgrup de 2-Sylow és correcte o no.

- **Càlcul de la valoració 2-àdica (valoracio_2-adica):** És una funció complementària que vam dissenyar per assegurar-nos que l'algoritme treballava correctament, és a dir, trobava el subgrup de 2-Sylow d'una forma correcta amb nombres petits. Ho he fet així perquè és una manera segura de provar que va bé; només es pot fer amb nombre petits ja que quan fem la valoració 2-àdica necessitem calcular el cardinal i, calcular aquest per a nombres grans, és un problema computacionalment difícil. Podriem dir que és una forma de comprovació dels resultats.

Paràmetres d'entrada: el cos de Galois $\mathbb{F}_{2^d}$, la corba el·líptica ja creada amb la corresponent equació i l'enter n que és 2-Sylow ja trobat.
Paràmetres de retorn: un booleà que ens dirà si la valoració és correcta.

Listing 4.9: Pseudocodi de la funció Valoració 2-àdica

```
enter resta_card, quocient_card;
cardinal(e);

divideix(quocient_card, cardinal, 2);
remainder(resta_card, cardinal, 2);

mentre resta_card:=0 fer
        divideix(quocient_card, cardinal, 2);
        remainder(resta_card, cardinal, 2);
        cardinal:=quocient_card;
        si resta_card:=0 llavors
                incrementa(n_copia);
        fsi
fmentre

si n_copia:=n llavors
        imprimeix(correcte);
        sino
        imprimeix(incorrecte);
        fsino
fsi
```

4.2.5 Paraules clau

En aquesta secció explicarem paraules que s'utilitzen dins al pseudocodi i que no són funcions, per tant aquestes seran funcions implementades dins la llibreria matemàtica LiDIA [5].

- **soluciona:** Correspon a la funció de LiDIA anomenada *solve_quadratic*, la qual ens crearà un polinomi amb els paràmetres passats i ens retornarà veritat o fals depenent de si té solució o no sobre el cos desitjat.
- **crea_cos:** Fa referència a la funció de LiDIA anomenada *galois_field*, que ens crearà el nostre cos de característica 2.
- **numero_elements_camp:** Fa referència a la funció anomenada *number_of_elements* que ens retornarà el número d'elements del nostre cos.

- **polinomi_irreductible:** Correspon a la funció *irred_polynomial* que ens retornarà el polinomi irreductible del cos.
- **aleatori:** Fa referència a la funció *randomize* i ens escollirà un número aleatori del cos per a crear la corba.
- **punt_pertany_corba:** Fa referència a la funció *on_curve* que ens dirà si el punt pertany o no a la corba (retornant un booleà).
- **comprovem:** Correspon a la funció *twice* i ens retornarà el punt entrat multiplicat per 2.
- **ordre:** Fa referència a la funció *order_point* i ens retornarà l'ordre del punt.
- **find_roots:** Correspon a la funció del mateix nom i ens trobarà les solucions existents del nostre polinomi.
- **crear polinomi dins el vector:** Assignem els coeficients a cada lloc on li pertoca dins de l'equació.
- **buscar_generador:** Correspon a la funció *generator*, la qual busca un generador del cos que ens buscarà tots els elements que existeixen menys el 0.
- **cardinal:** Fa referència a la funció *group_order* i ens retornarà el cardinal del grup.
- **controlem_errors:** És una serie de comprovacions amb funcions de LiDIA per comprovar que tot funciona correctament.
- **crear_corba:** Correspon a la funció en LiDIA anomenada *set_coefficients* que ens crearà la corba amb els coeficients que li passem.
- **crear_punt:** Fa referència a la creació del punt inicial i ho fem amb *assign* i li passem la x i la y.
- **inserta:** Correspon a la funció *insert_at* i insereix en un vector el valor passat, per després poder calcular el polinomi.
- **divideix:** Fa el que diu el mateix nom dividir dos valors *bigint*, correspon a la funció *divide*.
- **remainder:** Ens troba la resta de la divisió dels coeficients que li passem, correspon a la funció *remainder*.

4.3 Software i Hardware

En primer lloc, veurem el software utilitzat per realitzar el projecte:

- Plataforma Eclipse versió 3.2
- Compilador gnu/g++ versió 4.0.3
- Sistema Operatiu Ubuntu, Kernel 2.6.15-28-386
- Llibreria matemàtica LiDIA versió 2.2.0
- MiKTeX versió 2.7
- TeXnic Center versió 7.01 'Greengrass'
- Gnuplot versió 4.2
- Ghostscript versió 8.54
- Ghostview versió 4.8
- Paquet *PSTricks* (latest version)
- Paquet *Beamer* (diapositives)

En segon lloc, veurem les característiques del hardware on s'ha executat o, dit d'altra manera, és la plataforma on s'ha dut a terme el projecte:

- Intel Pentium IV
- Velocitat de processador 3GHz
- 512 MB de memòria RAM

Capítol 5

Resultats

En aquest capítol presentem les diverses proves que s'han realitzat per avaluar els diferents modes d'execució anteriorment explicats.

Les proves han consistit en generar corbes aleatòries i calcular-ne el 2-Sylow de cada corba generada; a més a més s'ha calculat el temps que es tarda a calcular el 2-Sylow de cada corba. També s'han realitzat proves buscant una corba específica i analitzant el temps utilitzat en trobar aquesta corba. Tot això acompanyat de gràfics, histogrames i taules per a comprendre-ho millor. S'han realitzat proves també per veure quin és el rendiment de la CPU en l'execució d'exemples de búsqueda de corbes.

També s'han executat les proves per veure, empíricament, que l'algoritme en què es basa aquest treball té un cost polinomial, que és el que s'esperava que fos. Les proves s'han realitzat en un Intel Pentium IV a 3GHz i amb 512MB de memòria RAM.

En les següents seccions es donaran a conèixer els resultats obtinguts en les diverses proves realitzades així com les conclusions a les que s'ha arribat després d'aquestes.

5.1 Resultats Obtinguts (modes 1 i 2)

En aquest primer apartat utilitzarem els modes d'execució 1 i 2. Hem realitzat una sèrie de proves en primer terme aleatòries amb els modes anteriors per veure i comprovar que realment l'algoritme funciona correctament i ens troba l'enter n que determina el subgrup de 2-Sylow corresponent a cada corba aleatòria. La diferència entre aquests dos modes rau en el fet que, en el primer mode, l'extensió del cos és elegida a l'atzar i en el segon mode l'extensió l'escollim nosaltres. Hem pogut observar que gairebé totes les corbes aleatòries elegides a l'atzar tenen un 2-Sylow amb $n = 1$, ja que cada cop que

creix l'extensió del cos creix també el número de corbes amb 2-Sylow $n = 1$, que en conclusió seran de les que n'hi ha més quantitat i de les que sempre n'hi haurà.

Veiem també que aquests 2 modes serveixen per trobar només una sola corba elegida a l'atzar entre moltes i ens retorna el 2-Sylow.

Paràmetres	Resultats
a_2	669286366961655426110986888675096929687961627103
a_6	13609117834423382100093217063394517371929535787 63
n	2
$t(ms)$	25
a_2	665191902092007210342690426887438151019065704397
a_6	183957036727189114190253790284499749246628714632
n	6
$t(ms)$	110
a_2	6324211346657350119200926341765462059491285192 22
a_6	740537728299053753321824603198972602281947613418
n	3
$t(ms)$	30
a_2	3542082813714524123751873571574553297022930595 94
a_6	141929846126568339344410038187769656917506243172 6
n	1
$t(ms)$	10
a_2	1424514607389654688551976408897986929904480112934
a_6	126059500289131710917391873196499144591812980176 3
n	1
$t(ms)$	10

Taula 5.1: Resultats en corbes sobre $\mathbb{F}_{2^{160}}$

5.2 Resultats Obtinguts (mode 3)

En aquesta primera part de resultats utilitzarem el tercer mode d'execució, al qual li havíem de passar dos paràmetres, el primer indicant-li el que volem fer 'calcular_corbes' i el segon un nombre. Aquest segon nombre serà l'extensió del nostre cos. Quan fem aquest tipus d'execució recomanem no ficar nombres molt grans ja que tardarà molt de temps en realitzar els càlculs. Nosaltres hem realitzat els càlculs per als cossos $\mathbb{F}_{2^d}$, des de $d = 2$ fins a $d = 10$.

El programa farà tots els càlculs i ens imprimirà per pantalla el número de corbes que hi ha per cada 2-Sylow i el temps emprat.

5.2.1 Anàlisi del Temps

Començarem primer veient el número de corbes que hem de calcular i el temps emprat depenent de l'extensió del cos d'entrada.

Extensió (d)	Número de corbes	Temps emprat (s)
2	16	0.01
3	64	0.07
4	256	0.13
5	1024	0.6
6	4096	3.08
7	16384	12.43
8	65536	66.24
9	262144	225.65
10	1048576	1023.81

Taula 5.2: Número de corbes a calcular depenent de l'entrada

En la taula anterior podem veure que per corbes petites l'algoritme és molt eficient ja que tarda molt poc temps, però a mida que el número de corbes va creixent el temps també. Veiem que per a 2^{10} tarda ja molt de temps en recórrer totes les corbes, 1023.81 s $\approx$ 18 min. També podem observar que cada vegada que l'extensió augmenta una unitat, el nombre de corbes es multiplica exactament per 4. Fet que es deu a què cada cop tindrem el doble del doble de corbes a recórrer: una pel primer bucle while del paràmetre a_2 i l'altra pel segon bucle while del paràmetre a_6 (vegeu pseudocodi funció trobar_n_maxima).

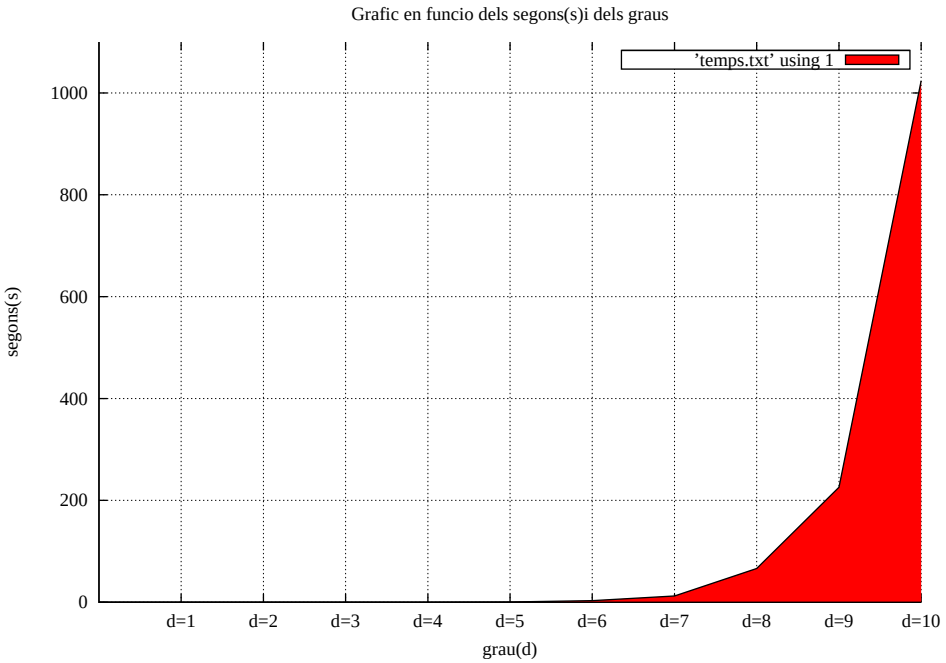


Figura 5.1: Evolució del temps

En aquest gràfic del temps podem veure com el temps es dispara quan l'extensió és 10. També hem de dir que aquest temps obtingut l'hem trobat fent la mitjana de 3 execucions de cada extensió i que aquest és el temps que tarda en recórrer totes les corbes i al qual s'hauria de sumar el temps que tarda en fer assignacions o crear el vector corresponent. Hem cregut que aquest temps no era necessari ja que el que ens interessava era trobar el temps que necessitàvem per trobar les corbes.

5.2.2 Anàlisi de les Corbes

Quan vam haver realitzat totes les proves per a totes les extensions de 1 fins a 10, vam obtenir totes les corbes vàlides (és a dir, que el discriminant no era 0). Per organitzar els resultats els hem col·locat en taules on els hem agrupat segons el cos $\mathbb{F}_{2^d}$ i el grup de 2-Sylow per poder-los classificar. Hem anat obtenint els resultats per $\mathbb{F}_{2^d}$, des de $d = 2$ fins a $d = 10$. Passem a veure i analitzar els resultats obtinguts i a observar què hi passa en les taules 5.3 i 5.4.

Com podem veure en les taules, ens apareixen resultats no esperats, per exemple quan el cas és $\mathbb{F}_{2^5}$. En aquest cas hi ha corbes amb grup de 2-Sylow $\mathbb{Z}_{2^n}$, amb $n = 3$ i $n = 5$ i en canvi no hi ha corbes amb $n = 4$.

2-Sylow (n) \ Extensió (d)	2	3	4	5	6
1	6	28	120	496	2016
2	4	16	64	256	1024
3	2	12	16	160	512
4	0	0	40	0	96
5	0	0	0	80	0
6	0	0	0	0	384
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0	0	0	0
10	0	0	0	0	0

Taula 5.3: Obtenció de totes les corbes de 2^2 fins a 2^6

2-Sylow (n) \ Extensió (d)	7	8	9	10
1	8128	32640	130816	523776
2	4096	16384	65536	262144
3	1792	9216	30720	131072
4	1344	4352	18432	66560
5	0	640	11520	30720
6	0	0	0	2560
7	896	0	0	0
8	0	2048	0	0
9	0	0	4608	0
10	0	0	0	30720

Taula 5.4: Obtenció de totes les corbes de 2^7 fins a 2^{10}

Aquests 0's ens van cridar molt l'atenció ja que no és normal que no ens apareguin corbes així com així d'un número determinat. Al final vam veure que aquests 0's es deuen a l'Interval de Hasse. En efecte, en aquest cas on el cos és $\mathbb{F}_{2^d}$ amb $d = 5$, l'Interval de Hasse és el següent:

$$I = \left[p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}\right]$$

$$I = [21.7, 44.3]$$

En aquest interval no hi ha enters de la forma:

$$m = 2^4 \cdot m', \text{ on } 2 \nmid m'$$

Com es pot veure a la taula 5.5:

2^1	2*12=24	Cau dins l'interval
2^2	4*6=24	Cau dins l'interval
2^3	8*4=32	Cau dins l'interval
2^4	16*3=48	NO cau dins l'interval
2^5	32*1=32	Cau dins l'interval

Taula 5.5: Taula de proves

Per tant hi ha corbes amb grup de 2-Sylow $\mathbb{Z}_{2^n}$, amb $n = 3$ i $n = 5$, però no n'hi ha amb $n = 4$.

Hem realitzat també uns histogrames de forma que els resultats obtinguts en les taules anteriors ens quedin ordenats. Hem fet histogrames de bloc on cada bloc es proporcional al número de corbes que té d'aquell 2-Sylow. Veurem que com anteriorment hem explicat hi haurà blocs que en moments puntuals no apareixeran, fet que és deu, com hem dit a l'Interval de Hasse, i voldrà dir que no hi haurà corbes d'aquell 2-Sylow. En aquests gràfics es veu que aproximadament el 50% de les corbes tenen $n = 1$ i que s'explica amb més detall en l'apartat de Percentatge de corbes (vegis Figura 5.2 i Figura 5.3).

En realitat aquests histogrames reflecteixen el que nosaltres obteníem aleatòriament amb els modes 1 i 2. Només hem fet les execucions per cossos $\mathbb{F}_{2^d}$, des de $d = 2$ fins a $d = 10$, ja que a partir d'aquesta extensió ja apareixia un número de corbes molt gran.

En els histogrames podiem veure els blocs ordenats segons el número de corbes i el grau de l'extensió del cos, en canvi en les figures 5.4, 5.5 i 5.6 podem veure com es distribueixen el número de corbes obtingut de cada cos, en cossos $\mathbb{F}_{2^d}$ amb $2 \leq d \leq 10$, respecte el 2-Sylow. Ho fem des

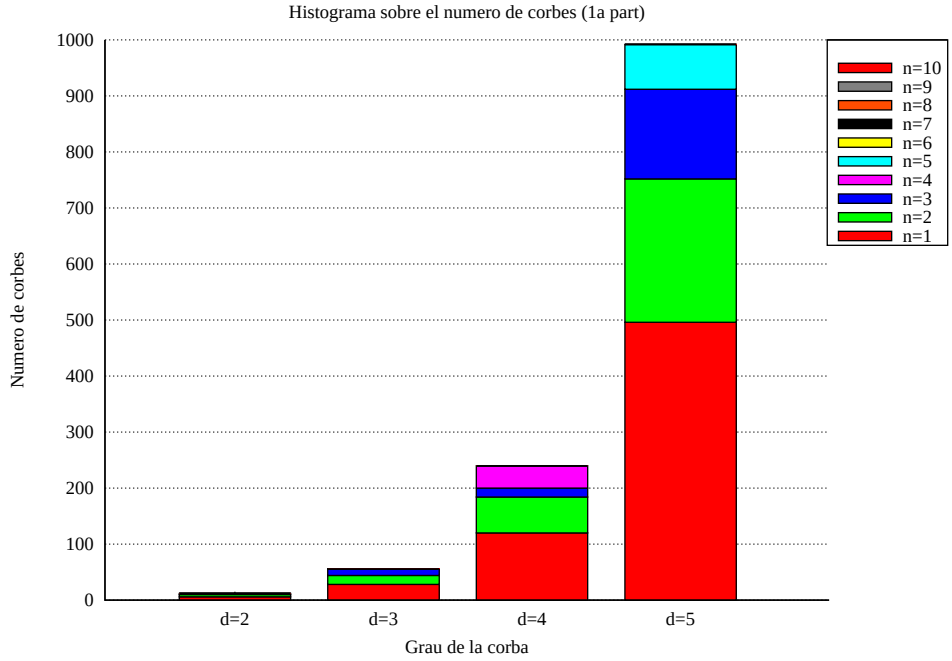


Figura 5.2: Nº de corbes sobre cossos $\mathbb{F}_{2^d}$ des de $d = 2$ a $d = 5$.

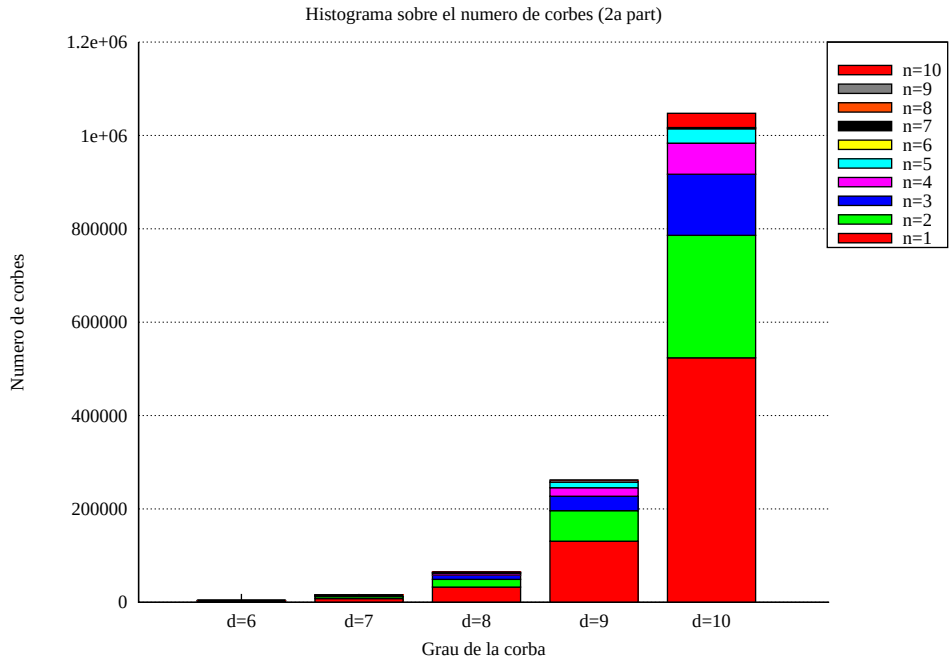


Figura 5.3: Nº de corbes sobre cossos $\mathbb{F}_{2^d}$ des de $d = 6$ a $d = 10$.

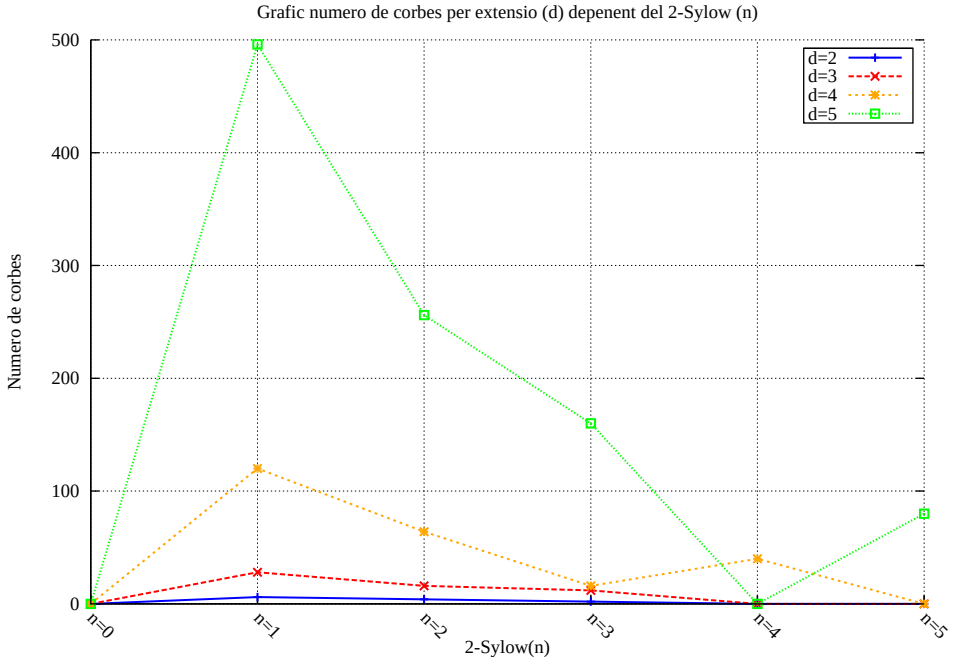


Figura 5.4: Nº de corbes sobre cossos $\mathbb{F}_{2^d}$ des de $d = 2$ a $d = 5$.

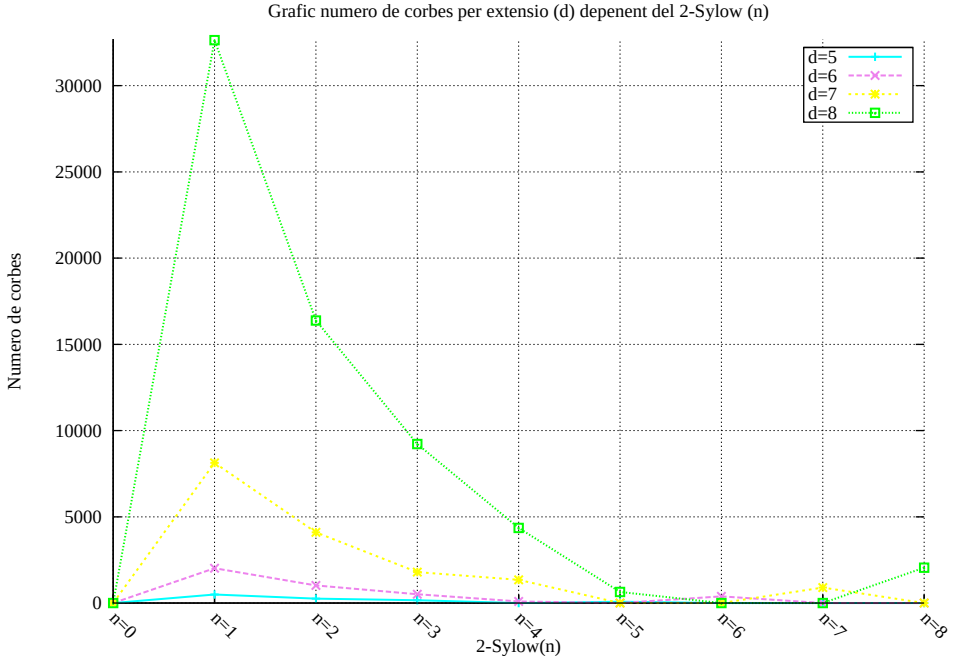


Figura 5.5: Nº de corbes sobre cossos $\mathbb{F}_{2^d}$ des de $d = 5$ a $d = 8$.

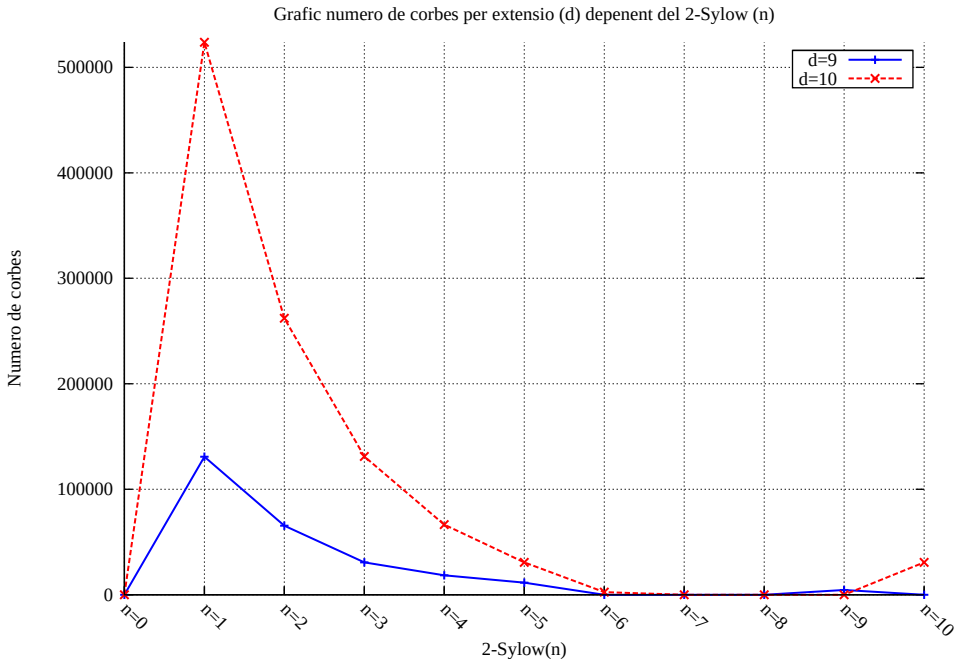


Figura 5.6: Nº de corbes sobre cossos $\mathbb{F}_{2^d}$ des de $d = 9$ a $d = 10$.

d'una perspectiva diferent a la mostrada anteriorment amb l'histograma, ara ho mostrem amb un diagrama de línies. També podem observar com ens apareixen els 0's.

Percentatges de les corbes

Al realitzar l'anàlisi de totes les corbes hem observat que hi ha un percentatge molt alt de corbes amb 2-Sylow de $n = 1$, $n = 2$ i $n = 3$. Per tant hem calculat quan era aquest percentatge per a poder predir si quan fem una execució amb els modes 1 o 2 amb una corba determinada, quin serà l'enter n del subgrup de 2-Sylow. Mirant les taules de l'apartat anterior hem calculat aquests percentatges en la taula 5.6.

Com podem veure a la taula amb $n = 1$, en tenim un 50%, encara que sembli curiós hem obtingut aquest nombre sense arrodonir. Per a $n = 2$ hem fet la mitjana de tota la columna i hem obtingut que la mitjana està al voltant del 26.6%, i per a $n = 3$ hem fet el mateix i hem trobat que la mitjana està sobre el 13.64%. Cosa que vol dir que si sumem aquestes tres mitjanes obtindrem que els tres primers 2-Sylows s'emporten pràcticament el 90% de les corbes i l'altre 10% és reparteix entre les altre corbes.

En la figura 5.7 podem veure com es mouen els resultats al voltant de la

2-Sylow (n) / Extensió (d)	1	2	3
2	50%	33.3%	16.6%
3	50%	28.57%	21.42%
4	50%	26.6%	6.6%
5	50%	25.80%	16.13%
6	50%	25.39%	12.69%
7	50%	25.19%	11.02%
8	50%	25.09%	14.11%
9	50%	25.04%	11.74%
10	50%	25.02%	12.51%

Taula 5.6: Percentatges de corbes amb 2-Sylow $\mathbb{Z}_{2^n}$ amb n=1, 2 i 3

mitjana, la línea verda representa la mitjana i les altres representen els valors reals que nosaltres hem obtingut. En la primera de les línies se sobreposen els resultats ja que els valors obtinguts són exactes, en canvi als altres si que es pot anar veient com varien. No hem fet el càlcul per n's més grans ja ens haguessin aparegut els 0's anteriorment explicats i els valors obtinguts no haguessin estat reals.

La conclusió d'aquests percentatges és que hi ha aproximadament la meitat de corbes que tenen 2-Sylow $\mathbb{Z}_{2^n}$ respecte les que tenen 2-Sylow $\mathbb{Z}_{2^{n-1}}$.

5.2.3 Anàlisi de la CPU

A l'hora de realitzar les proves em vaig fixar que la CPU sofria una acceleració repentina mentre calculava la nostra solució. Aquest fet que em va encuriosir i vaig decidir investigar-ho. Vaig fer un script que simplement utilitzava la comanda *top* la qual ens permet veure els processos què utilitzen la CPU i quin ús en fan en concret. Vaig obtenir cada 5 segons el rendiment total de la nostra CPU i vaig redireccionar-lo a un fitxer. Llavors mentre tenia activat aquest script vaig executar Eclipse (vegis secció 4.3) amb la nostra aplicació per a cossos $\mathbb{F}_{2^d}$, amb $d = 5$, 6 i 7 i vam veure que no estavem equivocats, el PC s'accelerava bruscament i la memòria també es consumia a un ritme alt. Una de les causes d'aquesta pujada pot ser que a l'interior de l'aplicació utilitzem vectors i segons les entrades que hi posem poden arribar a ser molt i molt grans. El script que hem dissenyat és molt simple i és així:

Listing 5.1: Línia de codi de la captura de CPU

```
#top -e -n100 | grep ^C >fitxer.txt
```

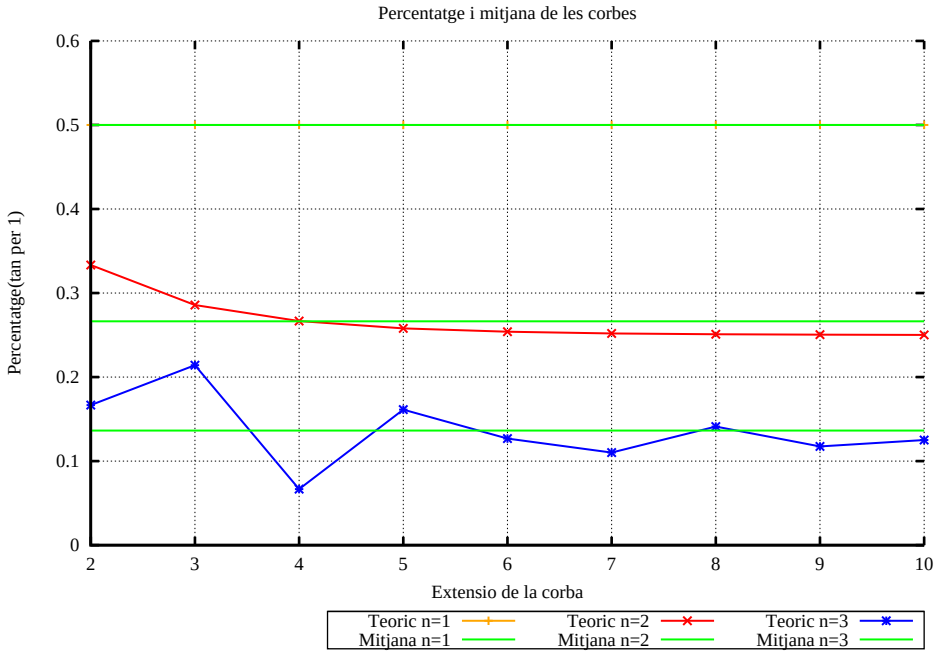


Figura 5.7: Percentatge de corbes segons el 2-Sylow

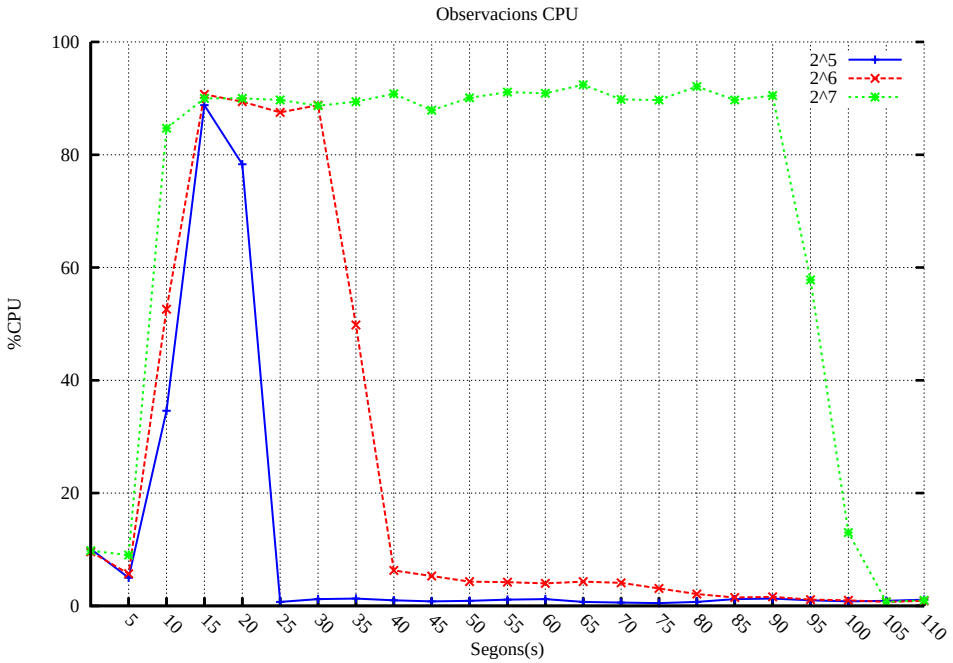


Figura 5.8: Acceleració CPU durant l'execució

Com podem veure en la figura 5.8 les acceleracions sempre arriben fins al 90% de la CPU, però mai arribaran al 100%, ja que és molt difícil trobar un programa que ens utilitzi tota la CPU d'una forma constant al 100%. També podem veure que de la mateixa forma que accelera, quan acaba es desaccelera ràpidament tornant a la normalitat i alliberant tota la memòria utilitzada per al procés.

5.3 Resultats Obtinguts (modes 4 i 5)

En aquesta secció hem realitzat proves amb els modes d'execució 4 i 5. Recordem que aquests dos modes servien per trobar corbes amb un 2-Sylow determinat dins un cos passat per paràmetres.

El mode 4 el podrem utilitzar en corbes on el 2-Sylow sigui més petit de 20. Ens analitzarà totes les corbes sobre aquell cos extensió i, quan hagi trobat una corba amb un 2-Sylow igual al que li hem passat, pararà. Hem de tenir en compte que en aquesta opció utilitzem vectors i els vectors tenen un límit i la nostra memòria RAM també.

Això significa que no podrem ficar nombres massa grans perquè llavors quan creem el nostre vector LiDIA ens dirà que és massa gran. Cal esmentar que estic treballant amb una RAM de 512MB i que segurament si tingués una RAM de 1GB o 2GB podria arribar uns quants nombres més amunt. Pot ser que si utilitzem aquest mode en nombres molt grans el nostre PC se'ns quedi bloquejat i haguem de reiniciar.

En canvi el mode 5 podrem utilitzar-lo indistintament tant si és una corba sobre un cos petit o gran. Ho podem fer així perquè aquest està basat en l'elecció de paràmetres completament aleatoris, sense utilitzar vectors. Per això quan executem aquest mode li hem d'introduir el número de corbes a executar ja que al haver-n'hi tantes tardaria molts dies en trobar-la.

També m'agradaria fer un incís molt important en l'aspecte del temps. Aquest algoritme implementat tarda temps en trobar la corba, degut a la quantitat de corbes que pot tractar sobre el cos i en canvi no tarda gens en trobar la solució d'aquella corba (ho troba en l'ordre de **ms**). Així que podem dir que l'algoritme tarda temps en trobar la corba però que quan la troba ja quasi ha trobat la solució. Ho podem veure amb alguns exemples a la taula 5.7.

Fixem-nos que per a cada extensió del cos hem anat a buscar un dels 2-Sylows més grans que hi podem trobar a dins, però en podem buscar qualsevol altre que vulguem. Ara bé, ens hem d'assegurar que caigui dins l'Interval de Hasse del cos extensió sinó no el trobarem.

Extensió (d)	2-Sylow (n)	Temps (resoldre la corba)
10	10	35 ms
15	15	45 ms
20	20	50 ms
25	25	60 ms
30	30	80 ms
35	35	90 ms
40	40	130 ms
45	45	150 ms
50	50	200 ms

Taula 5.7: Taula de temps sobre 2-Sylows grans

També cal dir que he buscat fins a un cos amb extensió $d = 50$ però ho hagués pogut allargar-ho fins quan hagués volgut.

Capítol 6

Conclusions

En aquest treball de final de carrera, hem dissenyat i implementat un algoritme que donada una corba el·líptica sobre un cos binari troba un enter n tal que el subgrup de 2-Sylow sigui isomorf a $\mathbb{Z}_{2^n}$. L'algoritme s'ha dissenyat seguint el mateix procés inductiu proposat per R. Moreno en [13] en el cas de corbes sobre cossos $\mathbb{F}_p$. Aquest era un dels objectius que en un principi ens haviem fixat. També hem vist que l'algoritme dissenyat té un cost polinòmic.

Cal comentar també que dins l'aplicació dissenyada hem implementat cinc modes d'execució. En un principi només en volíem un (que trobés el 2-Sylow i ja està), però a l'hora d'implementar vam veure que podíem fer una ampliació de les funcionalitats, quedant de la següent manera: la primera ens trobarà el subgrup de 2-Sylow d'una corba on el grau de l'extensió del cos és elegit a l'atzar, el segon dels modes farà el mateix però el grau de l'extensió li introduirem nosaltres, el tercer mode ens calcularà totes les corbes existents dins d'un mateix cos i els modes 4 i 5 ens serviran exclusivament per trobar 2-Sylows grans. La diferència entre tots dos rau en el fet que el mode 4 només trobarà 2-Sylow de l'ordre de 20 i el mode 5 trobarà 2-Sylow de l'ordre desitjat.

Un altre dels objectius que pretenïem assolir era el de trobar 2-Sylows grans (de 50 o més grans). Aquest també ha estat assolit i com hem pogut veure en l'apartat dels Resultats, ens tarda de l'ordre de milisegons en trobar el 2-Sylow (on s'inverteix més temps és en trobar la corba que compleixi amb les condicions, no el 2-Sylow).

En l'apartat de resultats també hem pogut veure el tema de les proporcions que hi ha entre el número de corbes, que és va mantenint encara que augmentem el grau de l'extensió del cos.

Hem pogut veure també en la realització i estudi d'exemples pràctics el fet que hi ha salts en la seqüència d'enters corresponent als 2-Sylows de les corbes sobre un cos binari determinat. Aquest fet s'explica per la forma de

l'Interval de Hasse.

Ja per acabar hem observat també que LiDIA ens comença a donar problemes a l'hora de calcular el cardinal, o bé l'ordre del punt per a cossos grans de l'ordre de 2^{200} ja que inverteix molt temps en càlcul per acabar a vegades sense fer-lo.

Finalment una última observació respecte la llibreria usada en el treball i és que *John Cremona* ens va comentar que en una versió futura de LiDIA (no la pròxima) s'intentaria corregir el bug trobat i explicat en l'apartat d'Implementació.

M'agradaria fer un incís també en què tot els projecte ha estat fet, dissenyat, document i exposat amb eines de software lliure i això per a un informàtic no hauria de ser res estrany sinó normal, ja que aquest és el futur de la informàtica.

Les línies futures d'aquest projecte de final de carrera podrien anar cap a l'estudi dels ℓ-Sylow, on ℓ sigui un primer més gran que 2, per a corbes el·líptiques sobre un cos de característica 2. També seria interessant i a la vegada laboriosa, l'opció del càlcul del ℓ-Sylow d'una corba el·líptica per a diferents primers ℓ petits. Ja s'han fet alguns treballs per a corbes sobre cossos de característica p.

Bibliografia

[1] A. Albajes. *Corbes el·líptiques: Un criptosistema semànticament segur i determinació de la 2-torsió*, pages 26–30. TFC, Universitat de Lleida, 2003.

[2] G. Belingueres. *Introducción a los sistemas de curva el·líptica*, pages 4–10. UNLP (Argentina), 2000.

[3] I. Blake, G. Seroussi, and N. Smart. *Elliptic curves in cryptography*, pages 19–20, 37–39. Cambridge University Press, 1999-2000.

[4] Certicom Corporation. Elliptic curve groups over finite fields. Disponible a: http://www.certicom.com/.

[5] LiDIA Group. *LiDIA-A library for computational number theory*. S.Hamdy Editions, Edition 2.1.1, May 2004.

[6] Wikipedia Group. Wikipedia. Disponible a: http://www.wikipedia.org/.

[7] P. Hewitt. *A brief history of elliptic curves*, pages 1–4. University of Toledo (USA), December 5, 2005. Disponible a: http://livetoad.org/Courses/Documents/132d/Notes/.

[8] C. Ivorra. *Curvas Elípticas*. Universitat de València, 2007. Disponible a: http://www.uv.es/ivorra/.

[9] N. Koblitz. *Elliptic curve cryptosystems*, pages no. 177, 203–209. Math. Comp. 48, 1987.

[10] Information Security Magazine. Rsa conference 2007. Disponible a: http://searchsecurity.techtarget.com/.

[11] A. Menezes. *Elliptic curve public key cryptosystems*, pages 83–90. Kluwer Academic Publishers, AH Dordrecht, The Nederlands, 1993.

[12] J. Miret, R. Moreno, A. Rio, and M.Valls. *Determining the 2-Sylow subgroup of an elliptic curve over a finite field*, pages 411–427. Mathematics of computation,74 no. 249, 2005.

[13] R. Moreno. *Subgrupos de Sylow de las curvas elípticas definidas sobre cuerpos finitos*. PhD Thesis, Universitat Politècnica de Catalunya, 2005.

[14] J. Ramió. *Introducción a la cifra con curvas elípticas*. Departamento de Publicaciones de la Escuela Universitária de Informática de la Universidad Politècnica de Madrid, 2006.

[15] R.A. Tomàs. *Determinación del grupo de 2-torsión de las curvas elípticas sobre cuerpos finitos*, pages 18–21. TFC, Universitat de Lleida, 2002.